D6.1 – Demonstrate scalability of global scale NWP ensemble at resolution of 5km exploiting EuroHPC pre-exascale computing resources

Deliverable ID	D6.1
Deliverable Title	Demonstrate scalability of global scale NWP ensemble at resolution of 5km exploiting EuroHPC pre-exascale computing resources
Work Package	WP6
Dissemination Level	Public
Version	0.4
Date	2022-08-31
Status	FINAL
Deliverable Leader	ECMWF
Main Contributors	

The ACROSS project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955648. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Italy, France, Czech Republic, United Kingdom, Greece, Netherlands, Germany, Norway.

Published by the ACROSS Consortium

Document History

Version	Date	Author(s)	Description
0.1	2022-06-27	ECMWF	First version of the ToC
0.2	2022-08-04	ECMWF	First version
0.3	2022-08-22	ECMWF, NP	Addressed reviewer's comments
0.4	2022-08-22	ECMWF	DAOS section, Benchmark results

Table of Contents

Glossary

Acronym	Explanation
API	Application Programming Interface
DB	Data base
FDB	Field DataBase
GRIB	General Regularly-distributed Information in Binary form
HPC	High Performance Computing
HPDA	High Performance Data Analysis
HSM	Hierarchical Storage Management
ICON	ICOsahedron Non-hydrostatic – Earth system model framework
IFS	Integrated Forecasting System
MARS	Meteorological Archival and Retrievable System
MPI	Message Passing Interface
NVMe	Non-Volatile Memory Express
NWP	Numerical Weather Prediction
NVRAM	Non-Volatile Random Access Memory
POSIX	Portable Operating System Interface
PMEM	Persistent Memory
RDMA	Remote Direct Memory Access
SLURM	Simple Linux Utility for Resource Management (job scheduler)
SSD	Solid State Disk
WMO	World Meteorological Society
WRF	Weather Research and Forecasting
WRFDA	Weather Research and Forecasting Data Assimilation

List of tables

List of figures

Executive Summary

Numerical Weather Prediction increasingly strives for higher resolutions in its simulations in space and, such, in time. Accordingly, the amount of data handled and produced by state-of-the-art Global NWP models is ever-increasing, posing an increasing challenge to the hardware and software used in this context. Whereas computing power has become cheaper and more readily available in recent decades, input-output infrastructures have to keep up with the data generated. Accordingly, weather and climate forecasting is one of the three pilot projects in the ACROSS project.
ECMWF operates a world-class Global-scale NWP model called IFS, and in the context of ACROSS is developing innovative solutions to improve its data management workflow, especially for large ensembles of experiments as well as storm-resolving model runs. This deliverable describes the meteorological and technological challenges and provides a detailed description of the innovations implemented and the corresponding results achieved on the EuroHPC computational resources.
An update of this deliverable is planned for M32, with the detailed description of all the implemented improvements and the NWP model execution results at scale.

Position of the deliverable in the whole project context

This deliverable is part of Work Package 6 “Weather, Climate, Hydrological and Farming Pilot”. In particular, it reports on the continued work undertaken in Task 6.1 “Data and metadata modelling for object-store integration” and Task 6.3 “Mesoscale weather simulations over the area of Eastern Mediterranean and Greek Peninsula” and it paves the way toward Task 6.4 “Complex Weather and Climate workflows at pre-exascale” that aims at demonstrating at scale the results achieved.
Thus, this deliverable directly contributes to the objectives of WP6:

1. Improve the existing operational system for global numerical weather prediction, post-processing and data delivery by exploiting hardware acceleration and data streaming/object-store techniques to demonstrate exascale scalability.
2. Enable low-latency exploitation of climate simulations by integrating data delivery through domain-specific object store
3. Develop and demonstrate an environment for user-defined in-situ data processing. The system will enable HPDA on multi-petabyte meteorological and climatological archives and data streams to enable data analytic workflows that improve insight to data.

Description of the deliverable

Deliverable D6.1 describes the efforts for improving the domain-specific object store (FDB) to efficiently exploit novel data stores available on the recent EuroHPC computational resources and provides measurable results achieved with each improvement
Section 1 focuses on the description of the IFS NWP model and the related challenges, with a focus on the data management requirements of operational exploitation of the model (Subsection 1.3)
Section 2 describes the innovations implemented and planned for the second half of the project, while Section 3 provides a description of the computational resources and an analysis of the results achieved.

1 Global-scale Numerical Weather Forecasts

Numerical weather prediction (NWP) uses mathematical models of the atmosphere and oceans to predict the weather based on current weather conditions. A number of global and regional forecast models are run in different countries worldwide, using current weather observations relayed from radiosondes, weather satellites and other observing systems as inputs.
The atmosphere is a fluid. As such, the idea of numerical weather prediction is to sample the state of the fluid at a given time and use the equations of fluid dynamics and thermodynamics to estimate the state of the fluid at some time in the future.
Mathematical models based on the same physical principles can be used to generate either short-term weather forecasts or longer-term climate predictions; the latter are widely applied for understanding and projecting climate change. This deliverable will focus on the short-term weather forecasts produced by the Integrated Forecasting System (IFS) global-scale NWP model developed by ECMWF

The rest of this section will describe the motivations for improving the Global-scale NWP, the main component of the IFS model and the workflow adopted in ACROSS WP6, Section 2 will focus on the improvements introduced in the context of the ACROSS project and Section 3 will describe the computational environment and the results achieved so far. An update is foreseen towards the end of the project.

1.1 Motivation

Manipulating the vast datasets and performing the complex calculations necessary for modern numerical weather prediction requires some of the most powerful supercomputers in the world. Even with the increasing power of supercomputers, the forecast skill of numerical weather models extends to only about six days. Factors affecting the accuracy of numerical predictions include the density and quality of observations used as input to the forecasts, along with approximations in the numerical models themselves. A more fundamental problem lies in the chaotic nature of the partial differential equations that govern the atmosphere. It is impossible to solve these equations exactly, and small errors grow with time (doubling about every five days). Present understanding is that this chaotic behaviour limits accurate forecasts to about 14 days even with accurate input data and a flawless model. In addition, the partial differential equations used in the model need to be supplemented with parameterizations for solar radiation, moist processes (clouds and precipitation), heat exchange, soil, vegetation, surface water, and the effects of terrain. In an effort to quantify the large amount of inherent uncertainty remaining in numerical predictions, ensemble forecasts have been used since the 1990s to help gauge the confidence in the forecast and to obtain useful results farther into the future than otherwise possible. This approach analyses multiple forecasts created with an individual forecast model or multiple models.
The combination of increased model resolution and ensemble forecasts is the main direction for improving the current state-of-the-art. To maximise the effectiveness of accurate weather forecasts, we have to produce, post-process and deliver the results in high-performance, low-latency workflows. Unfortunately, this requires the availability of huge computational resources and poses serious requirements on the data management subsystem.
Across is mainly focusing on innovations in the post-processing and data management components of the Global-scale NWP workflow, described in Section 2.

1.2 The Integrated Forecasting System (IFS)

The comprehensive Earth-system model developed at ECMWF in co-operation with Météo-France forms the basis for all data assimilation and forecasting activities carried on at ECMWF. All the main applications required are available through one computer software system called the Integrated Forecasting System (IFS) that is composed by several modules for simulating Atmospheric, Marine and Land components.

Atmospheric dynamics

Atmospheric dynamics deals with motion in the atmosphere and its thermodynamic state. ECMWF is leading the research on this subject and aims to improve the mathematical equations, numerical methods, and the dynamical core of the forecast model, as well as technical aspects such as implementation on high-performance computers.
The dynamical core in the forecast model discretises the Euler equations of motion, resolving flow features to approximately 4-6 grid-cells at the nominal resolution. The subgrid-scale features and unresolved processes are described by atmospheric physics parametrizations.

The dynamical core of IFS [1] is hydrostatic, two-time-level, semi-implicit, semi-Lagrangian and applies spectral transforms between grid-point space (where the physical parametrizations and advection are calculated) and spectral space. In the vertical the model is discretised using a finite-element scheme. A reduced Gaussian grid is used in the horizontal.
The IFS also has extra configurations available for research experiments that are not used operationally. An example is the non-hydrostatic dynamical core.
Hydrostatic equilibrium describes the atmospheric state in which the upward directed pressure gradient force (the decrease of pressure with height) is balanced by the downward-directed gravitational pull of the Earth. On average the Earth's atmosphere is always close to hydrostatic equilibrium. This has been used to approximate the Euler equations underlying weather prediction models and successfully applied in NWP and climate prediction. Non-hydrostatic dynamical effects start to become important below horizontal scales of about 10km. The current ECMWF model uses a hydrostatic dynamical core based on the spectral-transform approach for all forecasts. A non-hydrostatic extension developed by the ALADIN modelling consortium, which has been made available by Météo-France through the IFS/ARPEGE collaboration, is in use at ECMWF for research purposes. The IFS-FVM is an alternative non-hydrostatic dynamical core based on a finite-volume discretisation. In the context of ACROSS we will assess performances with both hydrostatic and non-hydrostatic dynamical core

Atmospheric physics

Atmospheric physics is a vital part of a weather forecast model and is often referred to as the physical parametrization. ECMWF research focuses on how to represent unresolved physical processes in the atmosphere, such as radiation, clouds and subgrid turbulent motions.

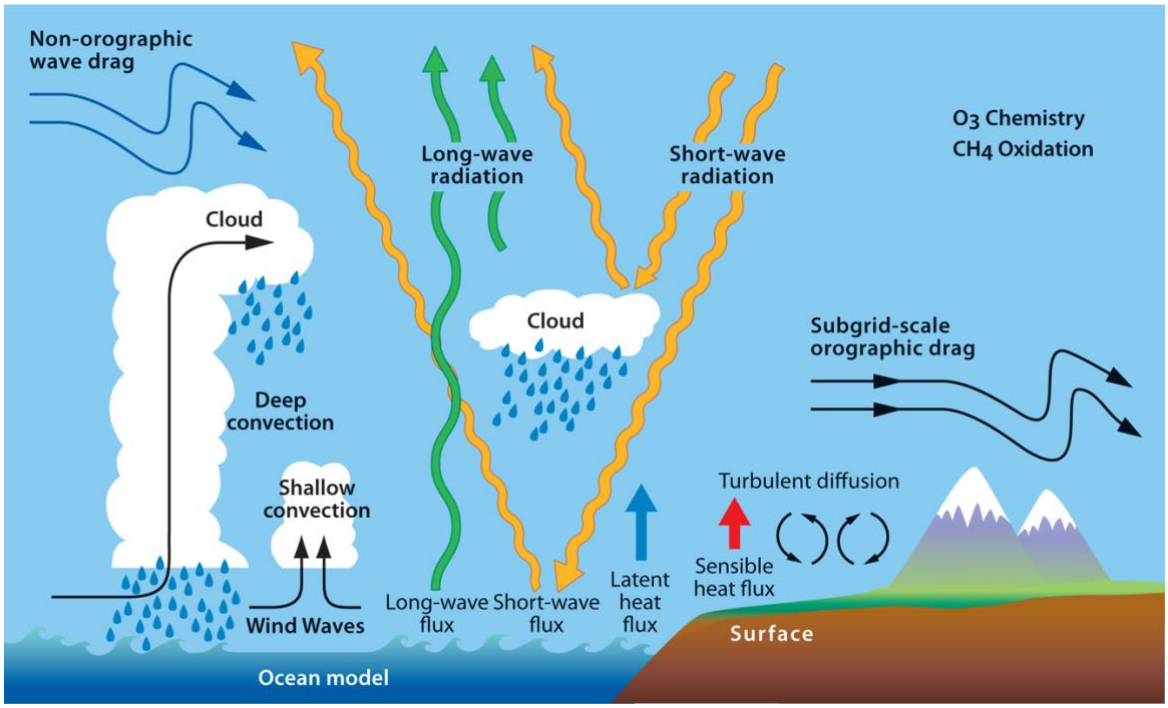

Figure 1 - Physical processes in the atmosphere modeled in IFS

The physical processes associated with radiative transfer, convection, clouds, surface exchange, turbulent mixing, subgrid-scale orographic drag and non-orographic gravity wave drag have a strong impact on the large-scale flow of the atmosphere. However, these mechanisms are often active at scales smaller than the resolved scales of the model grid. Parametrization schemes are then necessary in order to properly describe the impact of these subgrid-scale mechanisms on the large-scale flow of the atmosphere. In other words the ensemble effect of the subgrid-scale processes has to be formulated in terms of the resolved gridscale variables. Furthermore, forecast weather parameters, such as two-metre temperature, precipitation and cloud cover, are computed by the physical parametrization part of the model.

The radiation scheme ecRad [2] performs computations of the short-wave and long-wave radiative fluxes using the predicted values of temperature, humidity, cloud, and monthly-mean climatologies for aerosols and the

main trace gases (CO2, O3, CH4, N2O, CFCl3 and CF2Cl2). Cloud-radiation interactions are taken into account in detail by using the values of cloud fraction and liquid, ice and snow water contents from the cloud scheme using the Monte Carlo Independent Column Approximation (McICA). The solution of the radiative transfer equations to obtain the fluxes is computationally expensive, so depending on the model configuration, full radiation calculations are performed on a reduced (coarser) radiation grid and only every hour. The fluxes are then interpolated back to the original grid. Approximate updates to the fluxes are performed every model gridpoint and timestep to account for and high-resolution structure in surface temperature and albedo, and the variation of solar zenith angle between radiation calls.

The moist convection scheme is based on the mass-flux approach and represents deep (including congestus), shallow and mid-level (elevated moist layers) convection. The distinction between deep and shallow convection is made on the basis of the cloud depth (< 200 hPa for shallow). For deep convection the mass-flux is determined by assuming that convection removes Convective Available Potential Energy (CAPE) over a given time scale. The intensity of shallow convection is based on the budget of the moist static energy, i.e. the convective flux at cloud base equals the contribution of all other physical processes when integrated over the subcloud layer. Finally, mid-level convection can occur for elevated moist layers, and its mass flux is set according to the large-scale vertical velocity.

Clouds and large-scale precipitation are parametrized with a number of prognostic equations for cloud liquid, cloud ice, rain and snow water contents and a sub-grid fractional cloud cover. The cloud scheme represents the sources and sinks of cloud and precipitation due to the major generation and destruction processes, including cloud formation by detrainment from cumulus convection, condensation, deposition, evaporation, collection, melting and freezing. Supersaturation with respect to ice is commonly observed in the upper troposphere and is also represented in the parametrization.

The surface parametrization scheme represents the surface fluxes of energy and water and corresponding sub-surface quantities. The scheme is based on a tiled approach representing different sub-grid surface types for vegetation, bare soil, snow and open water. Each tile has its own properties defining separate heat and water fluxes used in an energy balance equation which is solved for the tile skin temperature. Four soil layers are represented as well as snow mass and density. The evaporative fluxes consider separately the fractional contributions from snow cover, wet and dry vegetation and bare soil. An interception layer collects water from precipitation and dew fall, and infiltration and run-off are represented depending on soil texture and subgrid orography [3]. A carbon cycle is included, and land-atmosphere exchanges of carbon dioxide are parametrized to respond to diurnal and synoptic variations in the water and energy cycles.

The turbulent diffusion scheme represents the vertical exchange of heat, momentum and moisture through sub-grid scale turbulence. The vertical turbulent transport is treated differently in the surface layer and above. In the surface layer, the turbulence fluxes are computed using a first order K-diffusion closure based on the Monin-Obukhov (MO) similarity theory. Above the surface layer a K-diffusion turbulence closure is used everywhere, except for unstable boundary layers where an Eddy-Diffusivity Mass-Flux (EDMF) framework is applied, to represent the non-local boundary layer eddy fluxes. The scheme is written in moist conserved variables (liquid static energy and total water). Convective clouds are treated separately by the shallow convection scheme.

The effects of unresolved orography on the atmospheric flow are parametrized as a sink of momentum (drag). The turbulent diffusion scheme includes a parametrization in the lower atmosphere to represent the turbulent orographic form drag induced by small scale (< 5 km) orography. In addition, in stably stratified flow, the orographic drag parametrization represents the effects of low-level blocking due to unresolved orography (blocked flow drag) and the absorption and/or reflection of vertically propagating gravity waves (gravity wave drag) on the momentum budget [4].

The non-orographic gravity wave drag parametrization accounts for the effects of unresolved non-orographic gravity waves. These waves are generated in nature by processes like deep convection, frontal disturbances, and shear zones. Propagating upward from the troposphere the waves break in the middle atmosphere, comprising the stratosphere and the mesosphere, where they exert a strong drag on the flow. The parametrization uses a globally uniform wave spectrum and propagates it vertically through changing horizontal winds and air density, thereby representing the wave breaking effects due to critical level filtering and non-linear dissipation.

Atmospheric Composition

Starting in the early 2000's, atmospheric composition has gradually become an increasingly important component of the IFS.
As part of the European Copernicus programme on environmental monitoring, greenhouse gases, aerosols, and chemical species have been introduced in the ECMWF model allowing assimilation and forecasting of atmospheric composition. At the same time, the added atmospheric composition variables are being used to improve the NWP system itself, most notably through the interaction with the radiation scheme and the use in observation operators for satellite radiance assimilation.

For the CO_2 modelling in the IFS, the land vegetation fluxes are modelled on-line by the CTESSEL carbon module [5]. The anthropogenic fluxes are based on the annual mean EDGARv4.2 inventory using the most recent year available with estimated and climatological trends to extrapolate to the current year. The ocean fluxes are from the climatology [6] and the fire fluxes are from GFAS. Because the budget is currently not constrained by observations, the CO_2 global bias can grow from one year to the next. In order to avoid this, the initial CO_2 forecast fields are updated every year with the most recent atmospheric 3D fields from the MACC-II CO_2 flux inversion system, whenever these become available, typically in October.
Methane fluxes are prescribed in the IFS using inventory and climatological data sets, consistent with those used as prior information in the CH_4 flux inversions from [7]. The anthropogenic fluxes are from the EDGAR 4.2 database. All the anthropogenic categories are based on annual mean values, except for rice, which has been modulated with a seasonal cycle. The wetland fluxes are from the Kaplan climatology [7]. The biomass burning emissions are from the MACC-II GFAS dataset. The other sources/sinks include wild animals, termites, oceans and a soil sink. For the chemical sink in the troposphere and the stratosphere, the climatological chemical loss rates from [8] are used.

The physical parameterizations dedicated to aerosol processes mainly follow the aerosol treatment in the LOA/LMD-Z model [9]. Five types of tropospheric aerosols are considered: **sea salt, dust, organic and black carbon, and sulphate aerosols**. Prognostic aerosols of natural origin, such as mineral dust and sea salt are described using three size bins. For dust, bin limits are at 0.03, 0.55, 0.9, and 20 microns; for sea salt, bin limits are at 0.03, 0.5, 5 and 20 microns.
Emissions of dust depend on the 10-m wind, soil moisture, the UV-visible component of the surface albedo and the fraction of land covered by vegetation when the surface is snow-free. A correction to the 10-m wind to account for gustiness is also included.
Sea-salt emissions are diagnosed using a source function based on [10].
Sources for the other aerosol types which are linked to emissions from domestic, industrial, power generation, transport and shipping activities, are taken from the SPEW (Speciated Particulate Emission Wizard), and EDGAR (Emission Database for Global Atmospheric Research) annual- or monthly-mean climatologies.
Several types of removal processes are considered: dry deposition including the turbulent transfer to the surface, gravitational settling, and wet deposition including rainout by large-scale and convective precipitation and washout of aerosol particles in and below the clouds. The wet and dry deposition schemes are standard, whereas the sedimentation of aerosols follows closely what was introduced by Tompkins [11] for the sedimentation of ice particles. Hygroscopic effects are also considered for organic matter and black carbon aerosols.

Atmospheric chemistry is fully integrated in the Integrated Forecasting System (IFS), complementing the aerosol modules and greenhouse gas variables for atmospheric composition. IFS with atmospheric chemistry supersedes a coupled system IFS-CTM, in which a global Chemistry Transport Model was two-way coupled to the IFS. The current system is computationally much more efficient than the IFS-CTM, and it avoids inconsistencies of the coupled approach.
The currently implemented chemical mechanism is an extended version of the CB05 chemical mechanism [12] as implemented in the Transport Model 5 (TM5), which describes tropospheric chemistry with 53 species and 107 reactions. Stratospheric ozone concentrations are currently parameterized with the Cariolle scheme.
The chemical solver used in C-IFS is the Euler Backward Iterative (EBI) solver [13]. The chemical time step employed is typically 15 minutes.
Dry deposition is simulated using pre-calculated dry deposition velocities. Wet deposition in C-IFS is based on the Harvard wet deposition scheme [14]. It accounts for sub-grid wet removal by considering cloud and precipitation area fraction. The input fields to the wet deposition routine are provided by the cloud scheme.
No emissions from lightning are a considerable contribution to the global atmospheric NOx budget. The estimate of the flash-rate density (flashes per time unit and area unit) is based on parameters of the convection

scheme. A global mass fixer is applied to ensure the mass conservation of the semi-lagrangian advection scheme.

Marine

The marine component of the Earth system has an important influence on the atmosphere on a range of timescales. We are developing a fully coupled model of the marine system including the surface waves, ocean and sea ice.
It has long been known that the waves affect the marine boundary layer of the atmosphere by modifying the surface roughness. All forecast systems at ECMWF are coupled to an ocean model. The ensemble and seasonal forecast systems use a coupled atmosphere-ocean model, which includes a simulation of the general circulation of the ocean and the associated coupled feedback processes that exist.
Ocean wave modelling
ECMWF uses and develops the WAve Model (WAM) model. It is coupled to the atmospheric model or run as a standalone model in the Limited-Area Wave (LAW) configuration.
Since 1998 ECMWF has been running a coupled forecasting system where the atmospheric component of the Integrated Forecasting System (IFS) communicates with the wave model (WAM) through exchange of the Charnock parameter which determines the roughness of the sea surface [15].
Our wave prediction system is based on a statistical description of oceans waves (i.e. ensemble average of individual waves). The sea state is described by the two-dimensional wave spectrum which gives the distribution of wave variance over different frequencies and propagation directions. Wave energy then follows from the product of water density, acceleration of gravity and wave variance.
The evolution of the wave spectrum follows from the energy balance equation which is explicitly solved by the WAM model. It determines the rate of change of the wave spectrum by adiabatic processes such as advection and refraction of wave energy, and by physical mechanism such as wind-wave generation, nonlinear four-wave interactions, and dissipation of energy by processes such as wave breaking, micro-scale breaking and bottom dissipation.
ECMWF uses the community ocean model NEMO (Nucleus for European Modelling of the Ocean) as part of the IFS. The NEMO model provides the dynamic ocean model used in the ensemble prediction system and the seasonal forecast system (S4). The ensemble prediction system of the medium and monthly range forecasts runs the ocean model at 0.25 degree horizontal resolution with 75 levels in the vertical and is initialised with the NEMOVAR (3D variational assimilation system) OCEAN5. The seasonal forecast system (S4) uses a 1 degree horizontal resolution with 42 levels in the vertical and is initialised with NEMOVAR OCEAN4.
Since 2013 the ensemble forecasts have coupled the atmosphere-wave-ocean model from the start of the forecast. This is important to allow capturing the two-way feedback between the atmosphere and the sea surface temperatures, for example when a tropical cyclone is slow moving it can cool the sea surface.
Since November 2014 the ensemble forecasts have been run with 0.25 degree horizontal resolution with the sea ice model active.
Sea ice is an important component of the Earth system; it is highly reflective, altering the amount of solar radiation that is absorbed; it changes the salinity of the ocean where it forms and melts, and it acts as a barrier to the exchange of heat and momentum fluxes between the atmosphere and ocean.
The current forecast systems model sea ice dynamically using the LIM2 (the Louvain-la-Neuve sea-ice model version 2) model within NEMO ocean model to represent the dynamic and thermodynamic evolution of sea ice within the coupled forecast system.

Land

The representation of soil, vegetation, snow, mountains, and water bodies is an integral part of the numerical weather prediction system at ECMWF. Land can affect the weather, the magnitude of the weather effects, and the evolution of human activities.
The effects of land surface state anomalies persist for several days and so raise the importance of a correct initial condition and modelled evolution. A refined representation of land surface processes and their accurate initialisation hold potential for further improvements in weather prediction up to monthly range as indicated in predictability studies.
Research on land surface state, therefore, spans the areas of parametrizations of biospheric and hydrological processes; the analysis techniques for the assimilation of satellite and in-situ observations; the land applications, including flood- and drought-monitoring and prediction, natural carbon fluxes, and fires.

1.3 Global-scale NWP workflow

The following is a diagram of the current ECMWF operational workflow, producing 9km HRES deterministic forecast and 18km 51-member ensemble forecasts. The key components are data acquisition and assimilation from heterogeneous data sources, high-performance integrated Earth system model, user defined product generation and push data delivery.

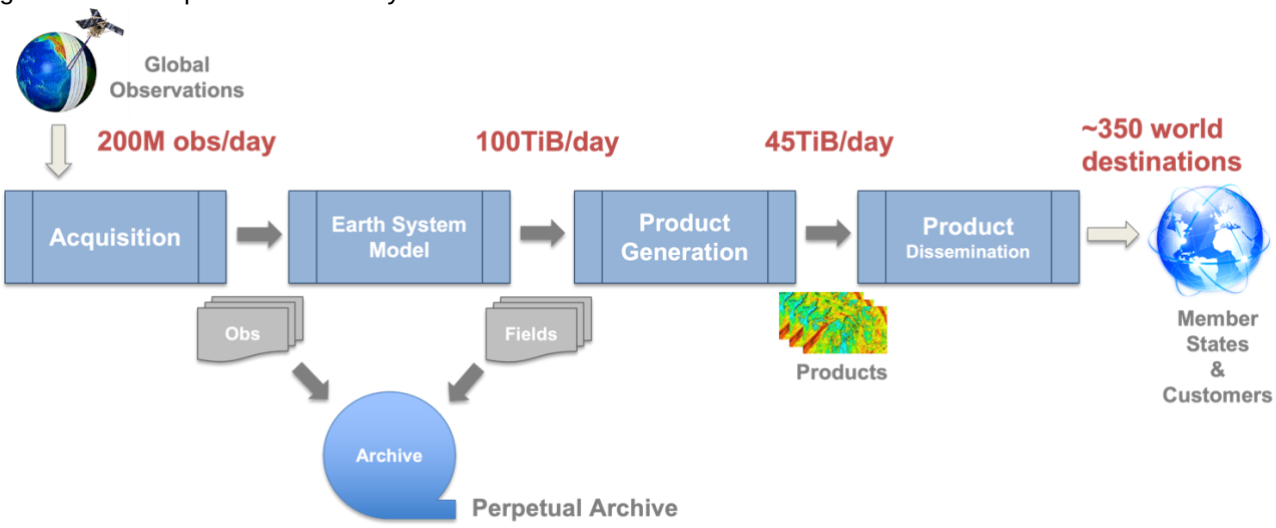

Figure 2 - High-level view of operational NWP workflow

From the data-management point of view, the critical aspect is the time-constrained execution of a large portion of the workflow in 1 hour window, with concurrent data producers (the numerical model) and data consumer (product generation) that, on average, reads back from disk 70% of the data just created.
This generates contention on the underlying data store.

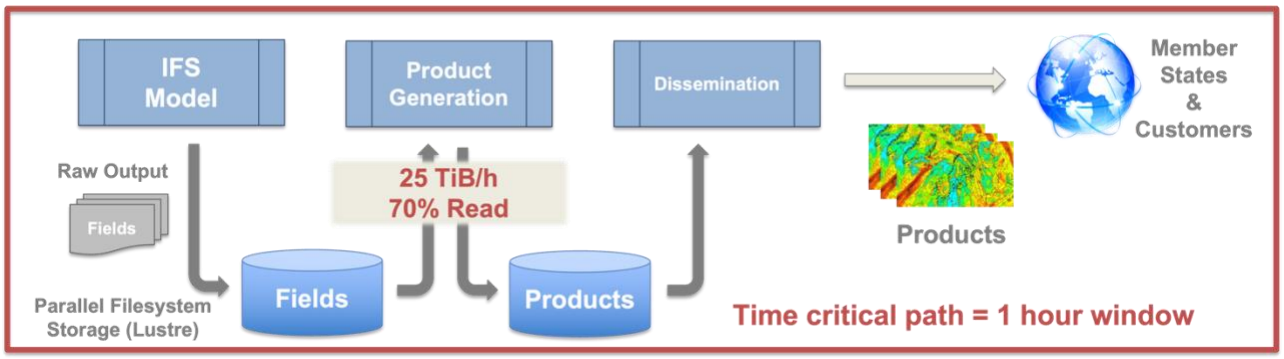

Figure 3 - Data management in the 1 hour time-critical window part of the workflow

The Global scale NWP workflow that we are developing as part of WP6 workflows, aims at demonstrating cloud-resolving simulations (4km) at global scale, with full post-processing workflow including user defined product generation. This is a 4.5x improvement over the current operational ensemble, and a brute-force approach will need more than 20x (4.5^2) computational and IO resources.
To achieve this result, we cannot simply rely on the hardware improvements in the I/O subsystem of the recent EuroHPC supercomputing resources and thus we have to develop innovations to streamline the data-management workflow and optimize the exploitation of novel data management subsystem such as Storage-Class Memory. The next section will detail all the innovations we have developed and we are currently working on.

2 ACROSS innovations in NWP

2.1 High-resolution IFS model

We aim at demonstrating IFS simulations at cloud-resolving resolution. In order to achieve this result on non-dedicated supercomputing resources, we have developed a self-contained version of the IFS numerical weather prediction system called Real Applications on Parallel Systems (RAPS), initially developed in 2012 in the context of the European Software Initiative. RAPS is continuously evolved to match the improvements introduced in the operational version of IFS as well as other experimental implementations (i.e. support for GPU accelerators).
In particular version 2.0 of RAPS can be compiled with OpenACC support and target execution on heterogeneous CPU+GPU computational resources.
The OpenACC optimization is focused two major improvements: support for spectral transform of the global-scale grid and GPU acceleration for microphysics scheme.

ECMWF's Integrated Forecasting System (IFS) is a global prediction system: entire earth's atmosphere is represented as a spherical grid. Info in "grid-point" space can be equivalently represented in "spectral" space, i.e. in terms of the frequencies of the fluctuating waves, which is more suited to some calculations.
IFS therefore repeatedly transforms between these representations, Fourier transforms (FFTs) in longitude and Legendre transforms (DGEMMs) in latitude, with AlltoAll data movement in-between.
The ESCAPE project focused on the development of optimized "dwarfs". In particular the Spherical Harmonics (SH) Dwarf represents the spectral transforms from IFS. The improvement consists of:

- Exposing Parallelism: Original implementations had naïve mapping of loops to the GPU, and the resulting decompositions did not map well. We have restructured to tightly nested loops, and used "collapse" OpenACC clause to allow compiler to map all inherent parallelism to hardware in an efficient manner.
- Optimizing data management such that the fields stay resident on the GPU for the whole timestep loop: all allocations/frees have been moved outside the timestep loop with temporary work arrays being re-used, and all host/device data transfer has been minimized.
- Memory Coalescing: Restructuring of array layouts to ensure memory coalescing. Sometimes transposes necessary: use OpenACC "tile" clause or push into BLAS library where possible.

The ECMWF Integrated Forecast System (IFS) cloud microphysics scheme has been adapted for a GPU architecture. Hybrid OpenMP and OpenACC within a single node, hybrid MPI and OpenACC over multiple nodes as well as different algorithmic and code optimization methods were employed to study the performance impact. The roofline model was used to conduct a performance analysis and the CLAW compiler has been explored as a tool for automatic code adaptation.

2.2 FDB Data management

Field DataBase (FDB) is a domain-specific object-store designed to handle meteorological datasets at scale and to cope with the compelling requirements of time-critical global-scale NWP and subsequent data post-processing.
Moreover FDB is based on a semantic indexing of the stored data: each data field is indexed by a key that is computed as a combination of the metadata associated with stored data. This is possible since we adopt self-describing data format, namely GRIB and ODB.

FDB architecture has recently been refactored to decouple the data storage part from the data indexing. The Catalogue is responsible for data indexing and data retrieval, while the Store handles bulk data archival.
The Store has been implemented with a modular architecture, in order to support different backends. Currently it can use both file systems and object storage. Each archived meteorological data object can be retrieved with a URI describing the file or object persisting the data and the offset of the relevant bytestream.

Once data are persisted by the Store, the Catalogue takes the responsibility of semantic classification and indexing by extracting metadata from the data header. The metadata are subdivided in three parts according with a user-defined hierarchy (defined in the FDB schema).
The first level identifies the so-called FDB *database*, the second level the FDB *index* and the third level the specific data object. The metadata used by the catalogue to uniquely identify a data object are called *Key*

FDB is designed to handle a search tree composed of 3 levels, but the metadata adopted in each level are fully user-defined in a regex-based schema file. Using the schema, we might decide to use date and time at

the top level for operational runs and push them to the second level for research data. Also, the set of relevant metadata is configurable and we support multiple definitions for different kinds of stored data.

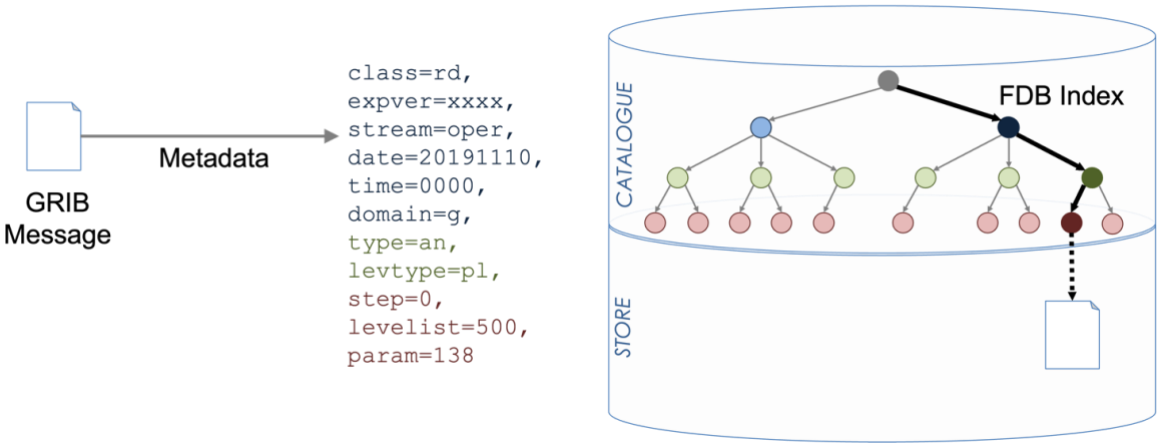

Figure 4 FDB archival with semantic indexing

2.2.1 Heterogeneous data store

We have already refactored our FDB in order to decouple the engine adopted for the index and the bulk data store. We have also preliminary experience with storage-class memory (i.e. Intel Optane DC). While we are confident to achieve the goal, demonstrating scalability at scale (we target roughly 1PB stored and indexed in 1hour time critical window, with low-latency post-processing on the same data) is a very ambitious goal that depends on FDB efficiency, resource availability and performance portability among the different computing resources targeted by the ACROSS platform.
We will benefit from the FDB modular structure to efficiently exploit the available data store in each supercomputing centre. In the case of multi-layer data stores, we are also developing a tool to migrate data from different data stores in order to keep the most recent (and thus more frequently accessed data) in the fastest storage and migrate older data to capacity layers.

2.2.2 IME

Infinite Memory Engine (IME) is a scale-out, software-defined, flash storage platform developed by DDN and available on CINECA Galileo100. It aims at decoupling the IO requirements of a diverse, concurrent set of HPC applications and the capabilities of large-scale parallel file systems. It acts as a caching layer: IME interfaces directly to applications and secures I/O via a data path that mitigates file system bottlenecks.

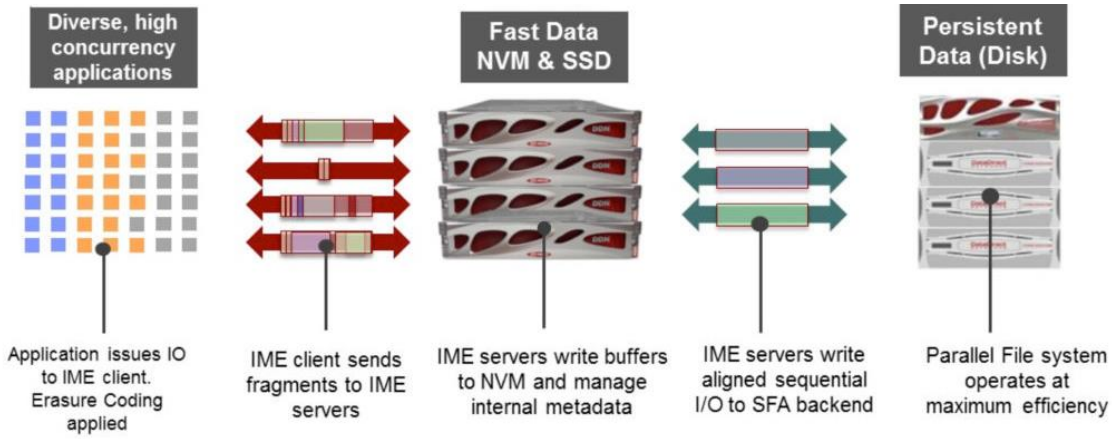

Figure 5 IME as a caching and IO optimization layer. Image courtesy of DDN

Along with increasing NAND capacities and the emergence of ultra-low latency, byte-addressable NVM, NVMe allows the low-latency potential of Non-Volatile memory devices to be realized across fabrics.
Traditional parallel file systems were developed when the underlying device latency was in the millisecond range when a thick layer of software incurring millisecond-order latencies was perfectly acceptable. Take that

same layer and introduce a flash backend and the fast file system becomes an IOPS barrier between application and storage media. This is even more critical when fast metadata handling is required.

In 2012 DDN started the development of IME to bridge this chasm between application and ultra-low latency storage, with a flash-optimized implementation that aims at longer lifetime for NAND flash.
In the context of FDB and IFS experiments, we aim at assessing the impact of IME both for the FDB catalogue and Store.

2.2.3 DAOS

The Distributed Asynchronous Object Store (DAOS) is an open-source high-performance object store designed for massively distributed NVM including SCM and NVMe. It provides a low-level key-value storage interface on top of which other higher-level APIs, also provided by DAOS, are built. Its features include transactional non-blocking I/O, fine-grained I/O operations with zero-copy I/O to SCM, end-to-end data integrity and advanced data protection. The OpenFabrics Interfaces (OFI) library is used for low-latency communications over a wide range of network back-ends.
DAOS is deployable as a set of I/O processes or engines, one per physical socket in a server node, each managing access to SCM (required) and NVMe devices within the socket. An engine partitions the storage it manages into targets to optimize concurrency, each target being managed and exported by a dedicated group of threads. DAOS allows reserving space distributed across targets in so-called pools, a form of virtual storage. A pool can host multiple trans- actional object stores called containers, each with their own address space and transaction history.
Upon creation, objects in a container are assigned a 128-bit unique object identifier, of which 96 bits are user-managed. Objects can be configured for replication and striping across pool targets by specifying their object class. An object configured with striping is stored by parts, dis- tributed across targets, enabling concurrent access poten- tially via multiple network adapters.

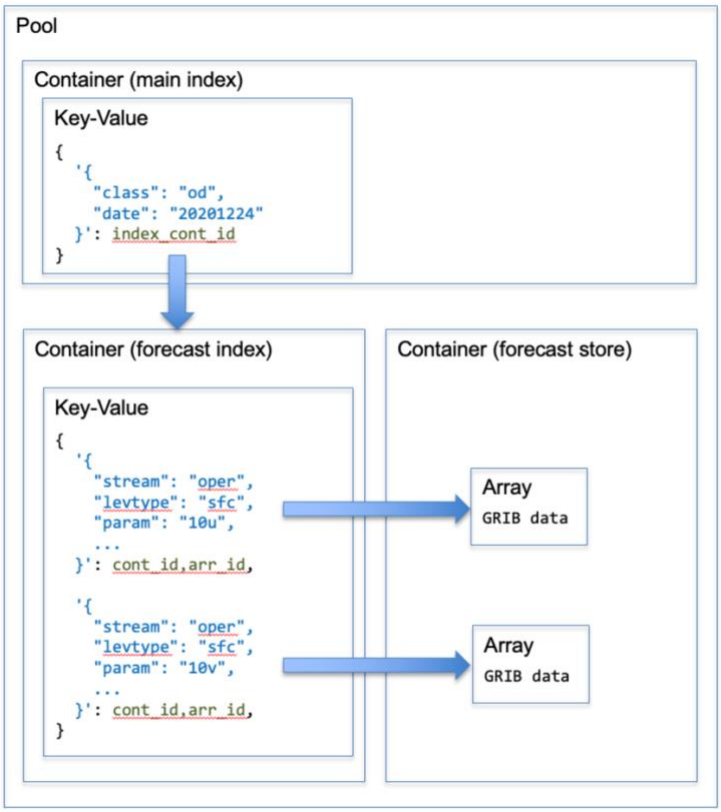

Figure 6 Diagram of DAOS concepts and APIs involved in the indexing and storage of weather fields.

A pair of C functions have been developed to perform writing and reading of weather fields to and from a DAOS cluster, using the DAOS C API. They have been developed based on the design of the FDB5 domain-specific object store already employed at ECMWF so that the same type of operations as in operational workflows is carried out. The diagram in Fig. 6 shows the different DAOS concepts and APIs involved in the field I/O functions.

At the top layer a DAOS Key-Value object, in its own container, acts as a main index telling where to find the data belonging to a same model run or forecast. That index maps the most-significant part of the key identifying a field (e.g. "'class': 'od', 'date': '20201224'") to another DAOS container, at the lower layer. In that container is another Key-Value object that acts as a forecast index, telling where to find the data of the fields comprising a forecast. That index maps the least-significant part of the field key to yet another DAOS container and a DAOS Array in that container. In that Array, the data of the indexed weather field is stored.
How the developed functions perform field writes and reads using these concepts and objects is described, in a very simplified way, in Algorithms 1 and 2, respectively.

Algorithm 1: field write

Inputs: field key, field data

```
open main container
query most significant part of field key from Key-Value
if not found then
        create forecast index and store containers
        register forecast index container in main Key-Value
end
open forecast store container
write field data into new Array
open index container
update forecast Key-Value
```

Algorithm 2: field read

Inputs: field key
Outputs: field data

```
open main container
query most significant part of field key from Key-Value
if not found then
    fail
end
open forecast index container
query least significant part of field key from Key-Value
if not found then
    fail
end
open store container
read field data from Array
```

When the field write function is called, with the binary field and its key as parameters, the most-significant part of the key is retrieved from the main Key-Value. If it exists, the indexed references to the forecast containers are retrieved and the containers are opened. If they do not exist, creation of a new pair of forecast index and store containers is attempted, with container IDs computed as md5 sums of the most-significant part of the key so that any concurrent processes attempting creation of the same pair of containers will avoid creation of inaccessible containers. Immediately after creation, the forecast containers are opened and the ID of the forecast store container is stored in a special entry in a newly created Key-Value in the forecast index container. Next, a reference to the forecast index container is indexed in the main Key-Value, using the most-significant part of the field key as key. Then, the binary field is written into an Array in the forecast store container, with a new object ID, and a reference to it is indexed in the forecast index Key- Value, using the least-significant part of the field key as key.
Note that when a field is written with a key that already exists in the overall store, a new Array object is created and indexed, and the previously existing one is de-referenced. No read-modify-write is performed upon re-write, and the functions do not delete de-referenced objects by design.
When the field read function is called, with the key of a field as the only parameter, the most-significant part of the key is looked up in the main Key-Value. If it exists, the indexed reference to the forecast containers is retrieved and the containers are opened. Next, the least-significant part of the key is retrieved from the forecast

index Key-Value. If it exists, the indexed references to the forecast store container and Array are retrieved. Finally, the forecast store container is opened and the Array is read.
For the NWP applications we are investigating, the separation of the different Key-Values and Arrays in different containers is motivated by the need to avoid contention for the same container or indexing Key-Value during intensive I/O workloads. This separation also allows for the implementation of the different parts of the object store with different storage back-ends or on different systems.

For the time being, the prototype is not yet integrated into FDB. We are currently working on the development of an FDB module based on such architecture. A noticeable advantage of the implementation as FDB backend is that we can combine a DAOS-based catalogue with different FDB store, in order to efficiently support configurations with limited SCM/NVMe resources.

2.3 In-situ data post-processing

The post-processing part is going to benefit from several innovations introduced in the ACROSS project and instrumental to the innovation of ECMWF operational workflow. In particular we aim at streamlining all the steps required to efficiently and reliably execute the product generation.

Product generation is implemented adopting a broker-workers pattern: a broker reads the user-defined requirements, derives the relevant input data and the required computational tasks, and dispatches the tasks to the workers.

For the time being, the steps required to post-process IFS output are:

1. IFS NWP model produces meteorological data fields (floating point, double precision)
2. GRIB encoding (simple packing to 16/24 bit accuracy + optional entropy compression)
3. GRIB messages are indexed and stored in FDB
4. As soon as data are available on FDB, the orchestrator (kronos) triggers the product generation
5. PGEN brokers analyse the user requirements and dispatches the tasks to PGEN workers
6. Each PGEN worker fetches from FDB the relevant data that are decoded to floating point, double precision.

At first we aim at improving PGEN data retrieval: the broker becomes responsible for fetching the data and dispatching data + tasks to the PGEN workers. This will reduce the load on the FDB and will contribute to reducing contention.

A second, more significant innovation, will enable in-situ post-processing: IFS output fields, after GRIB encoding, are stored to FDB and simultaneously dispatched to PGEN workers directly by MultIO middleware. FDB is no longer loaded by PGEN reads. The resulting steps are:

1. IFS NWP model produces meteorological data fields (floating point, double precision)
2. GRIB encoding (simple packing to 16/24 bit accuracy + optional entropy compression)
3. MultIO, according to user requirements, dispatches data + tasks to PGEN workers
4. Each PGEN worker decodes the GRIB data and process them.

We are also considering a further improvement: the dispatch of raw meteorological fields to PGEN workers, thus removing the overhead of GRIB encoding and decoding. This last step is only possible in case of in-situ post-processing, but has consequences on the product generation: we cannot guarantee bit-perfect reproducibility. In fact, in-situ post-processing is going to use full 64-bit floating point representation, while a re-run must still fetch encoded data from FDB, thus with potential small differences in the output data.

3 Computational infrastructure

3.1 CINECA Galileo100

3.1.1 Computational resources

The CINECA Tier-1 infrastructure GALILEO100, co-funded by the European ICEI (Interactive Computing e-Infrastructure) project and engineered by DELL has been installed in September 2021.

It has two main HPC computational partitions, complemented by a cloud partition (not exploited in our experiments):

1. 528 computing nodes equipped with 2x Intel CascadeLake 8260, 24 cores, 2.4 GHz, 384GB RAM, subdivided in:
    - 348 standard nodes ("thin nodes") with 480 GB SSD
    - 180 data processing nodes ("fat nodes") with 2TB SSD, 3TB Intel Optane
2. 36 GPU nodes with 2x NVIDIA GPU V100 with 2TB SSD.

3.1.2 Data stores

20 PB of storage accessible from both cloud and HPC nodes, capable of 120GB/s
720 TB fast NVMe storage (DDN Infinite Memory Engine solution) capable of 515 GB/s
540 TB of Storage Class Memory (Intel Optane DC)

We will use the heterogeneity of the available data stores to assess the benefits of each technology for data management requirements of high-resolution numerical weather forecasts. In particular we will assess the impact of each technology either for metadata and large data storage.

3.2 IT4I Karolina

3.2.1 Computational resources

The petascale system Karolina, acquired as part of the EuroHPC Joint Undertaking, reaches a theoretical peak performance of 15.7 PFlop/s. It was installed in 2021 ranking 69th worldwide in the TOP500 list and 8th in the Green500 list of the most energy-efficient supercomputers.

The supercomputer consists of several parts. We will exploit the two main computational partitions:

3. an universal partition for standard numerical simulations (720 nodes), with a peak performance of 3.8 PFlop/s
   720 nodes with 2x AMD 7763 CPUs, 64 cores, 2,45 GHz, 9,216 cores in total
4. an accelerated partition with 72 servers and each of them being equipped with 8 GPU accelerators providing a performance of 11.6 PFlop/s for standard HPC simulations and up to 360 PFlop/s for artificial intelligence computations,
   72 nodes with 8x NVIDIA A100 GPU each, 576 GPUs in total

3.2.2 Data stores

Karolina offers the following data stores:

- A redundant NFS file system for user homes (30.6 TB, 1.93 GB/s sequential write performance, 3.10 GB/s sequential read performance)
- A large high-performance Lustre parallel file system configured as $SCRATCH (1,361 TB) fully based on NVMe with 730.9 GB/s sequential write performance and 1,198.3 GB/s sequential read performance
- 3 high-capacity file systems (5PB each) based on NFS over GPFS, shared with the others IT4I computational resources.

Our experiments on Karolina are based on the High-performance $SCRATCH file system

3.3 Benchmark results

3.3.1 Lustre

To assess Parallel File System performances, we have developed a tool, called fdb-hammer, designed to use real GRIB data and stress PFS write performance together with FDB indexing capabilities. It mimics the load produced by a writing process (I/O server) within a forecast simulation.
This tool takes a sample GRIB file containing one field, and repeatedly resets the metadata keys on this GRIB before writing the contents to the configured FDB. The data written in each iteration is unchanged.
The sintax is:

```
fdb-hammer [--statistics] [--read] [-nensembles=] [--number=<n>]
  --nsteps=<nsteps> --nlevels=<nlevels> --nparams=<nparams> --expver=<expver>
  --class=<class> <grib_path>
```

With the following options:

--statistics	Report statistics after the run
--read	Read rather than write the data
--nsteps=integer	Number of steps. Data will be flushed after each step
--nensembles=integer	Number of ensemble members. If specified, GRIB supplied must support keyword number.
--nlevels=integer	Number of levels
--nparams=integer	Number of parameters
--expver=string	Expver to set on the data
--class=string	Class to set on the data
--number=integer	The first ensemble member number to use

In our tests, we adopted a GRIB field of 3.2MiB, a variable number of IO servers (2,4,8 or 16) and a variable number of IO threads for each server (from 1 to 32).

IO server	threads	time	Throughput	
(num)	(num)	(s)	GiB/s	TiB/h
2	1	253.23	1.89	6.64
2	2	133.71	3.58	12.58
2	4	68.81	6.95	24.44
2	8	39.47	12.12	42.60
2	16	25.08	19.07	67.04
2	32	27.70	17.26	60.69
4	1	127.51	3.75	13.19
4	2	65.06	7.35	25.85
4	4	34.26	13.96	49.08
4	8	20.37	23.48	82.55
4	16	17.14	27.90	98.10
4	32	15.31	31.23	109.80
8	1	83.72	5.71	20.09
8	2	42.24	11.32	39.81
8	4	21.78	21.96	77.21
8	8	11.50	41.60	146.25
8	16	8.52	56.15	197.41
8	32	7.96	60.12	211.36
16	1	42.19	11.34	39.86
16	2	21.18	22.58	79.40
16	4	11.04	43.30	152.24
16	8	6.19	77.21	271.43
16	16	10.08	47.43	166.74

Table 1 Lustre performances

The results achieved are summarized in Table 1, suggesting that scalability with respect to threads is almost linear up to 16 threads, which is also the sweet spot in our test case.

3.3.2 IME

IME is designed to be transparent from the application point of view: no code changes are required. We thus adopted the same Lustre testing procedure. Results are reported in Table 2
All our writes are explicitly flushed to disk, and this is probably the reason write performances to IME are essentially identical to writes to Lustre (with a little additional latency in most cases)

IO server	threads	time	Throughput	
(num)	(num)	(s)	GiB/s	TiB/h
2	1	223.26	2.14	7.53
2	2	119.87	3.99	14.03
2	4	69.96	6.84	24.04
2	8	39.52	12.10	42.55
2	16	30.28	15.79	55.53
2	32	53.19	8.99	31.61
4	1	134.39	3.56	12.51
4	2	82.56	5.79	20.37
4	4	43.98	10.88	38.24
4	8	24.10	19.84	69.77
4	16	16.82	28.43	99.94
4	32	26.89	17.79	62.53
8	1	78.73	6.08	21.36
8	2	42.25	11.32	39.80
8	4	22.85	20.93	73.57
8	8	12.98	36.84	129.52
8	16	9.50	50.34	176.97
8	32	14.31	33.43	117.52
16	1	39.85	12.00	42.20
16	2	21.88	21.86	76.86
16	4	11.85	40.37	141.93
16	8	7.11	67.27	236.51
16	16	6.99	68.40	240.46

Table 2 IME performances

3.3.3 DAOS

To assess the performance of DAOS, different I/O work- loads have been run: the well-known IOR benchmark and the field I/O functions have been employed to generate and run the workloads. The performance of DAOS and the benchmarks in the HPC system has been analysed based on different throughput definitions.

IOR
IOR is a community-developed I/O benchmark which relies on MPI to run and coordinate several parallel processes per- forming I/O operations against a storage server. It includes back-ends to operate with various popular storage servers, and DAOS is one of them.
IOR has been employed in this analysis with the intent to measure the throughput an application could achieve if it were programmed to operate as IOR and the DAOS back-end do. That is, running a number of parallel client processes which use the high-level DAOS Array API to write or read data from a DAOS cluster, all of them starting each I/O operation simultaneously, and waiting for each other to finish.
For the tests in this analysis, the IOR benchmark has been run in ***segments*** mode. IOR is invoked with a set of parameter values which instruct each client process to perform a single I/O operation, transferring its full data size. This is with the intent to assess the performance of a hypothetical parallel application which is designed to minimise the number of I/O operations and any operation interacting with the storage, as traditional parallel file I/O often does. Processes in such application issue a single trans- fer operation comprising all the data parts they manage, in contrast to an equivalent, non-optimised application where processes issue a transfer operation or even an open and a close operation for each data part. Unless the storage is not optimised

to handle large transfers or objects, this benchmark mode should give an idea of what is the maxi- mum, ideal throughput the storage can deliver. This mode has been implemented by setting both the -b and -t IOR parameters to the size of each data part managed by each process, and -s to the number of data parts managed by each process. The -i parameter has been set to 1, and the `-F` flag (file per process) has been enabled. All in all, each process performs the following:

1. initial barrier,
2. pre-I/O barrier,
3. object create/open of size `t * s` bytes,
4. transfer (write or read) of `t * s` bytes,
5. object close,
6. post-I/O barrier,
7. post-I/O processing and logging,
8. final barrier.

Field I/O

To gain insight into how DAOS performs in the previously outlined NWP I/O server use case, we use a custom field I/O benchmark which mimics operational NWP I/O workflows. Here, parallel processes perform each a sequence of field I/O operations with the functions described in Weather field I/O, with no synchronisation. Pool and container connections in a process are cached.

In contrast to the IOR benchmark in segments mode, the processes in this benchmark perform multiple I/O operations of smaller size, one for each data part. Each field I/O operation involves an Array open, transfer and close, and usually involves operations with Key-Values. We also do not enforce process or I/O start synchronisation, as in IOR, and there is no intermediate processing between I/O operations.

The field I/O benchmark has been configured in three different modes in the analysis.

- full: the full-blown field I/O functions are used as described in Weather field I/O.
- no containers: the use of container layers in the field I/O functions is disabled with the intent to analyse any potential flaws or performance-degrading aspects in the use of container layers. The functions create and operate with all DAOS Key-Values and Array objects in the main container.
- no index: the use of indexing Key-Value objects and container layers in the field I/O functions are disabled with the intent to compare with results from other benchmark modes and assess the overhead of field indexing. To preserve the ability to locate previously written fields, the field I/O functions map the field identifiers to DAOS Array object IDs by calculating an md5 sum of the identifiers. The arrays are stored and read directly from the main container.

By default, in modes with indexing enabled, each parallel process writes and reads fields indexed in its own forecast index Key-Value and therefore very low contention should be generated for the different objects within DAOS. The benchmark, however, can be configured to have a single shared forecast index Key-Value among all processes, inducing maximum contention on that Key-Value.

The IOR and field I/O benchmarks, in each of their modes, can be further adjusted and employed as part of larger workflows to generate specific combinations of I/O workload of interest, hereafter referred to as access patterns. Two access patterns used in this analysis are described next.

A (unique writes then unique reads):

This access pattern has two phases. Each client process performs a number of writes to different new objects in the storage (single transfer to single object for IOR in segments mode). Once all writer processes on all nodes have terminated, a second phase begins, where another process set of the same size and distribution is executed, each process performing the same number of reads from storage of the corresponding objects written in the first phase. The access pattern ends when all second-phase processes terminate.

This access pattern is designed to provide a situation where there is no contention for same fields and when there is only either write or read workload in operation at any one time, analogous to a single large-scale application utilising the object store.

In this access pattern, when run with the IOR benchmark, there is no contention in Array writes or reads, as each process accesses its own Array independently from oth- ers. When run with the field I/O benchmark in any of its modes with a forecast index Key-Value per process, there is no contention in any of the forecast index or store objects, as each process writes or reads its own Key-Value and Arrays sequentially.

B (repeated writes while repeated reads):

This access pattern starts with a setup phase to populate the storage with data to be used in the next phase. Half of the client processes (and thereby half the client nodes) performs a single write to a new object in the

storage. Once all writer processes on all nodes have terminated, the main phase begins. Half the client processes perform a number of re-writes to designated objects; simultaneously, the other
half of the client processes performs the same number of reads from their designated objects. The access pattern ends when all main-phase processes terminate.
This access pattern has only been implemented with the field I/O benchmark as IOR does not allow coordination between a set of writers and a set of readers, and aims to mimic situations where one or more large-scale applica- tions issue simultaneous writes and/or reads of the same fields. This is the equivalent of the I/O behaviour of actual NWP and product generation workloads.
In this access pattern, when run with Field I/O in full or no containers modes with a forecast index Key-Value per process, there is no contention in Array writes or reads, but there is some contention in each forecast index Key- Value between reader and writer processes on the same object. When run in no index mode, the same degree of contention occurs at the Array level.

IOR performances
For a single server, the bandwidth achieved is of approximately 2.5 GiB/s for write and 5 GiB/s for read. As the number of servers increases, the bandwidths increase linearly at a rate of approximately 2.5 GiB/s write and 3.75 GiB/s read for every additional engine.
It is worth noting that configurations with twice as many client nodes generally deliver best performance, both for write and for read. Setups with a substantially higher ratio (e.g. four times as many) of client nodes do not show significant increases in performance, and configurations with a slightly lower ratio show a substantial decrease in performance.
Above 8 server nodes, the scaling rate seems to decrease slightly even if using twice as many client nodes, as seen in the bandwidth results for the setup with 10 server nodes and 20 client nodes.

Figure 7 depicts the results achieved with up to 16 DAOS servers and up to 20 clients.

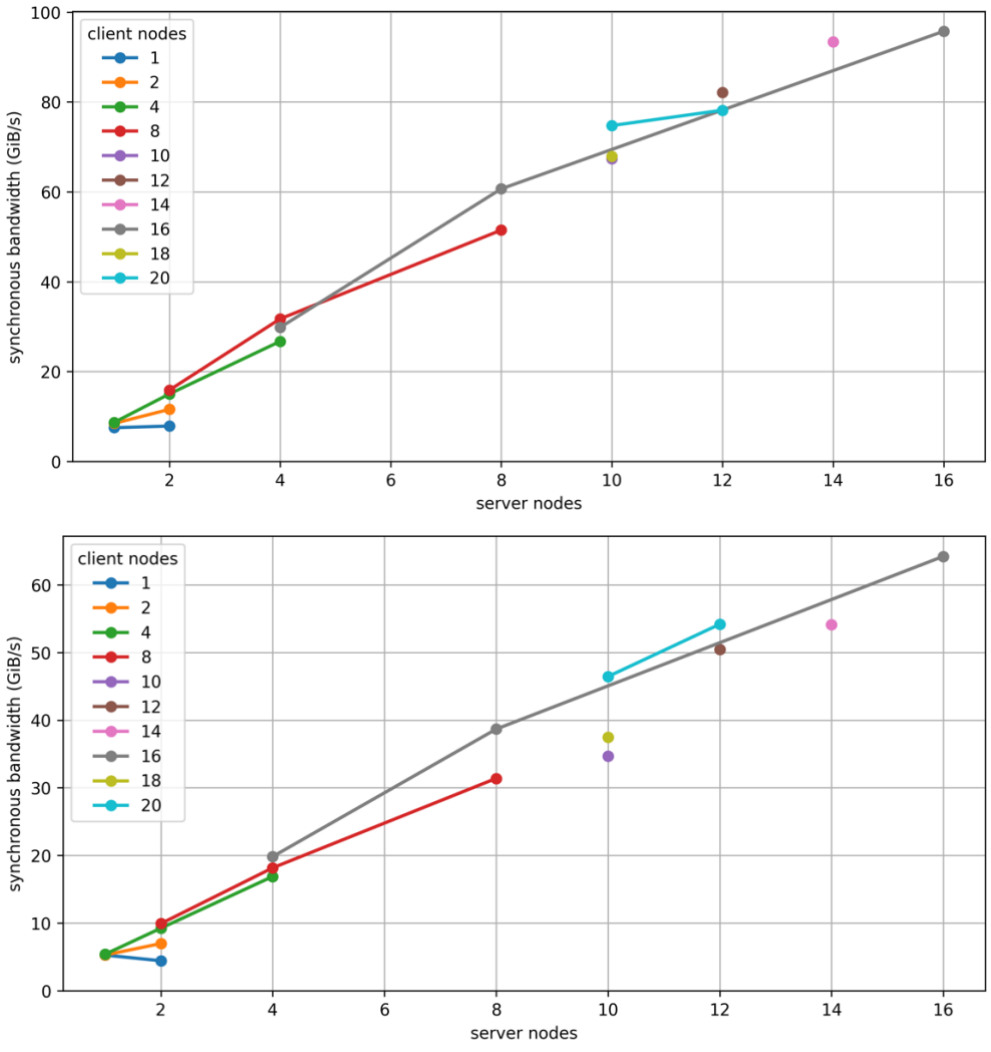

Figure 7 Mean synchronous write and read bandwidth results for access pattern A| (unique writes then unique reads) with IOR DAOS in segments mode.

Field I/O results

In this test, the Field I/O benchmark has been run with the intent to analyse the order of magnitude and behaviour of bandwidths obtained with the field I/O functions rather than simple segment transfers; how the different field I/O modes scale; and how bandwidth behaves in access pattern B with an I/O workload similar to operational workloads.

Access patterns A and B have been run with all modes of the Field I/O benchmark, configured for maximum contention on the indexing Key-Values. DAOS has been deployed on varying numbers of server nodes (1 to 8), with two engines per node. Benchmarks have been repeated 10 times, using the same varying numbers of client nodes as in IOR tests, and varying processes per client node (1 to 144). The averaged results for the best configurations are shown in Fig. 8.

The Key-Value objects have been configured with striping across all targets (OC_SX), and no striping (OC_S1) has been configured for the Array objects. This configuration was chosen on the assumption that Key-Values can be accessed by multiple processes at a time for these benchmarks and could therefore benefit from striping across targets, whereas Arrays are never accessed by more than one pro- cess simultaneously.

The number of I/O operations per client process has been set to 2000, and the I/O size to 1 MiB, resulting in Array objects of 1 MiB. The iteration count of 2000, higher than the segment count of 100 used in IOR tests, is necessary due to the lack of synchronisation in Field I/O, to reduce the effect of any process start-up delays in global timing bandwidth measurements.

The bandwidths obtained are in the same order of magnitude as those observed with IOR in segments mode. They are generally lower, as expected, due to the increased complexity and amount of storage operations involved. In a few cases the bandwidths are higher, and this is likely due to the large object and transfer sizes involved in IOR in segments mode, which may impact negatively IOR bandwidths.

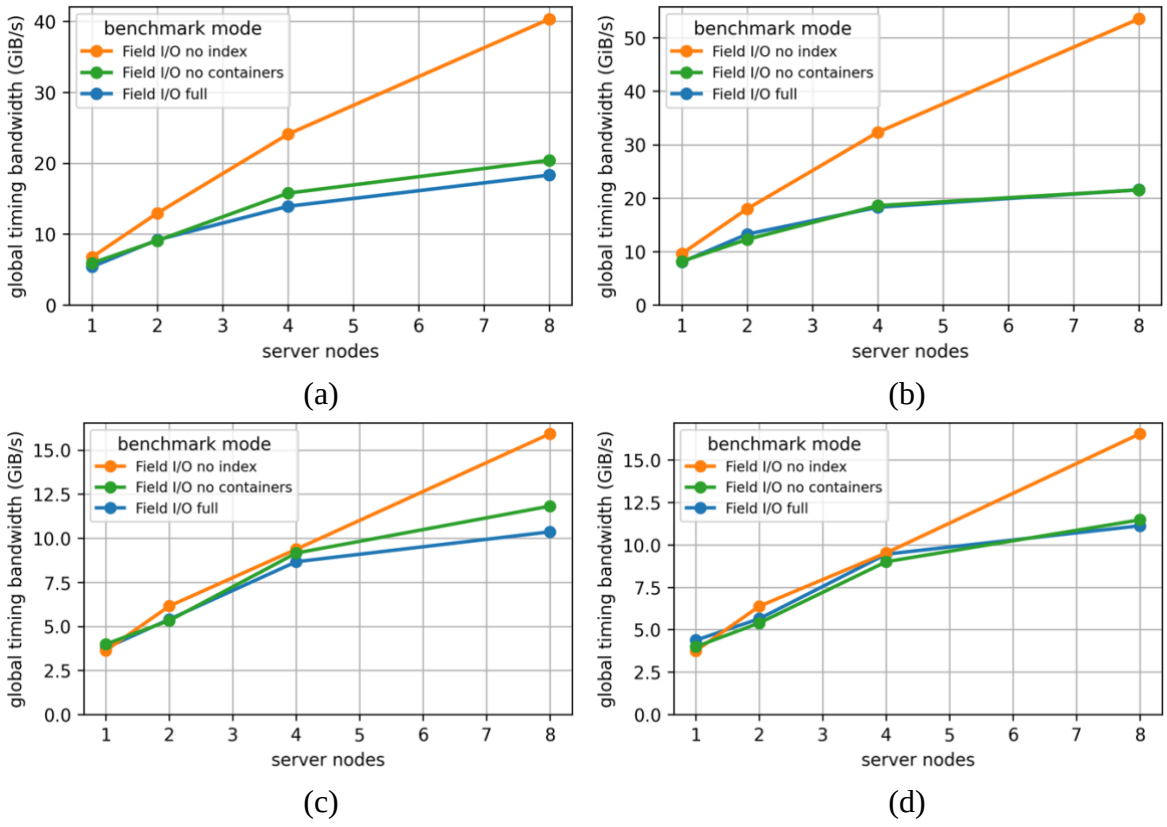

Figure 8 Global timing write (a) and read (b) bandwidth results for access pattern A (unique writes then unique reads, with high contention on the Key-Value objects) with the Field I/O benchmark, with high contention on the Key- Value objects. Global timing write (c) and read (d) for access pattern B (repeated writes while repeated reads)

The graphs demonstrate that bandwidth scales as the number of server nodes is increased, even with the high degree of contention, with all Field I/O implementations and access patterns, with a slightly better scaling in access pattern B.
It can seem that access pattern B performs significantly worse than pattern A as the bandwidth reached in the former is less than the half in the latter. However, note that in access pattern B the writers run simultaneously with the readers, and therefore the write and read bandwidth should be aggregated as an approximate measurement of overall application throughput. It can be observed, if calculating the aggregated bandwidths, that there is no substantial performance degradation in access pattern B compared to access pattern A, which is another encouraging result for the NWP use case.
The simplified mode of the Field I/O functions without indexing (no indexing Key-Values or containers) scales better than the two other implementations, at a rate of approximately 2.5 GiB/s for write and 3.75 GiB/s for read, per engine, in access pattern A. This scaling rate is similar to that observed with the IOR benchmarks and demonstrates the capability of DAOS to perform and scale well even when, in contrast to IOR segments, separate I/O operations are issued for the different data parts managed by the client processes. In access pattern B, the increase in aggregated bandwidth for that mode is approximately 2 GiB/s per engine, which we consider to be very good performance given the read/write contention.
For the modes with indexing, we observe the use of DAOS containers does not have a substantial impact on Field I/O performance in this scenario. These modes scale at a rate of approximately 3 GiB/s of aggregated bandwidth for pattern A, and 2 GiB/s of aggregated bandwidth per engine in pattern B, however, the scaling rate decreases beyond 4 server nodes down to 0.5 GiB/s of aggregated bandwidth per engine.
The results shown so far have been obtained with the Field I/O configured with a single shared forecast indexing Key-Value, inducing very high contention. This is however a very pessimistic scenario, and such degree of contention is unlikely to occur in operational workloads. The complete test set has been re-run with lower contention, where each client process uses its own forecast index Key-Value, to mimic an optimistic usage scenario. The results, obtained for the same server and client node counts as in IOR results, are shown in Fig. 9.
In these figures, configurations where the number of client nodes employed was less than twice the number of server nodes, have been represented with hollow dots. For the rest of the configurations, where twice the amount of client nodes or more were employed, are marked with solid dots.

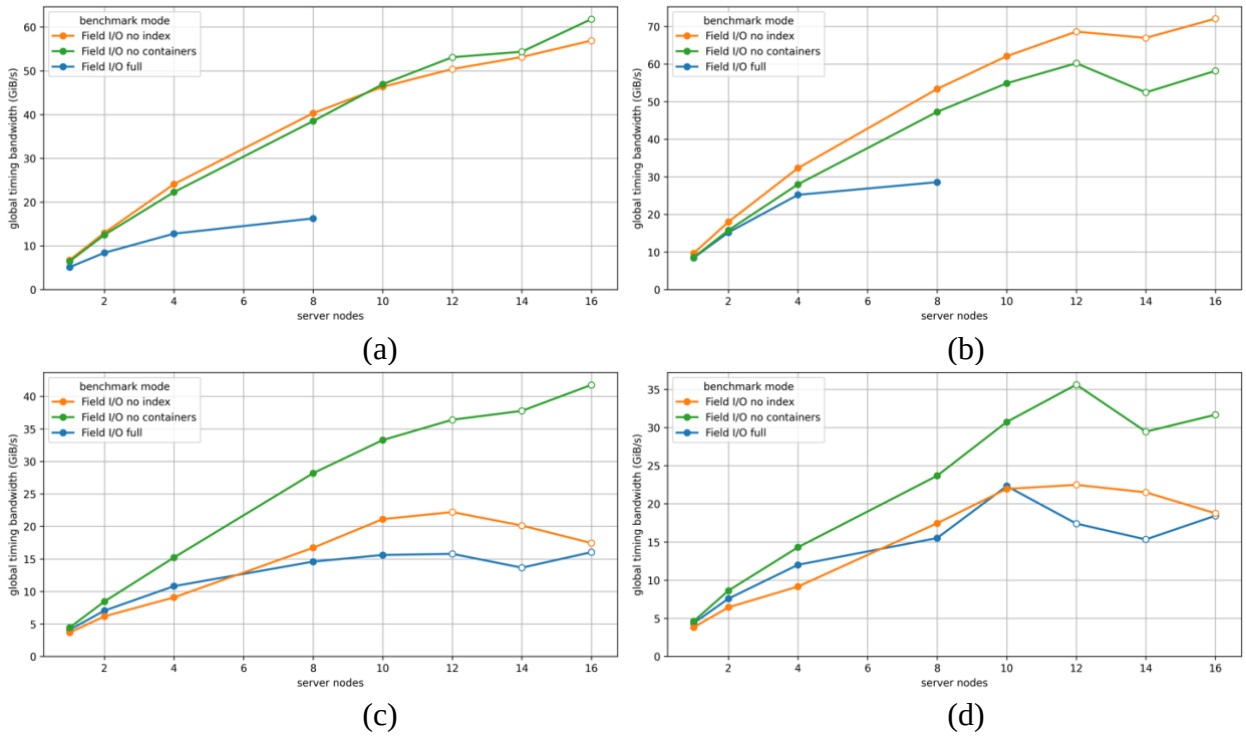

Figure 9 Global timing write (a) and read (b) bandwidth results for access pattern A (unique writes then unique reads, with high contention on the Key-Value objects) with the Field I/O benchmark, with low contention on the Key- Value objects. Global timing write (c) and read (d) for access pattern B (repeated writes while repeated reads)

The results for access pattern A show that the Field I/O implementation without containers scales remarkably well along with the mode with no indexing, particularly for write where, for larger server node counts, the mode with indexing performs better than the mode without indexing. This is encouraging and helps demonstrate the suitability of the object storage approach for the operational weather forecasting workloads we are evaluating here. The full mode scales at a similar rate or slightly better than in pattern A with high contention.
For access pattern B, the mode without indexing and the full mode scale at a rate of approximately 1.6 GiB/s of aggregated bandwidth per additional engine. Beyond 10 server nodes, both modes start to show a decline in performance.
The mode without containers stands out, scaling at a rate of approximately 2.75 GiB/s of aggregated bandwidth per additional engine. With this mode, employing 12 server nodes, a total aggregated application bandwidth of approximately 70 GiB/s is achieved. These encouraging results demonstrate again the potential of object storage and DAOS for the NWP use case.
Further work will be necessary to investigate the cause of the low performance obtained with the Field I/O mode with containers, to address any flaws or optimise the use of DAOS containers.

Conclusions

This deliverable offered the description of the challenges associated with the execution of Global-scale Numerical Weather Prediction at cloud-resolving resolution, a description of all the innovations planned and implemented in order to improve the Data Management subsystem adopted in the IFS model and as a backbone of ACROSS WP6 workflows. We have also analysed the effect of the implemented innovations on EuroHPC computational resources.

The effort spent optimizing FDB paid of, and we managed with as little as 16 IO server running 8 thread each to achieve >250 TiB/h. Our expectation is that IFS runs at cloud-resolving resolution (4km) will generate about 180TiB for each ensemble model run, thus the results achieved so far suffices.

We also managed to assess the impact of DDN IME subsystem, and in a write-only scenario, with full data flush enforced by the data management software system, the IME is not providing any speed-up. In preliminary tests on mixed read-write data access patterns, IME has a beneficial effect in reducing the IO contention. Further investigations are planned in the upcoming months.

The DAOS prototype showed good scalability and extremely relevant potential, but also clarified that the number of DAOS servers dedicated to NWP IO must be substantial (>16) and the number of clients has to be tuned to match the number of servers (we achieved the best results when the number of clients is roughly double the number of servers). Further analysis with full DAOS-FDB integration will be provided.

This document will be updated during the project to reflect any updates or changes that arise regarding the pilot. During the next period, the pilot partners will focus their efforts in the realization of the pilot goals and KPIs that have been set

References

There are no sources in the current document.

[1] Hortal, M. (2002) *The development and testing of a new two-time-level semi-Lagrangian scheme (SETTLS) in the ECMWF forecast model.* Q. J. R. Meteorol. Soc, 128, 1671–1687.

[2] Hogan, R. J., and A. Bozzo (2018) *A flexible and efficient radiation scheme for the ECMWF model.* J. Adv. Modeling Earth Sys., 10, 1990-2008.

[3] Balsamo, G., et al. (2009) *A revised hydrology for the ECMWF model: Verification from field site to terrestrial water storage and impact in the Integrated Forecast System.* J. Hydrometeorol., 10, 623-643.

[4] Lott, F. and Miller, M. J. (1997) *A new subgrid-scale orographic drag parametrization: Its formulation and testing.* Q. J. R. Meteorol. Soc., 123, 101-127.

[5] Boussetta, S., et al. (2013) *Natural carbon dioxide exchanges in the ECMWF Integrated Forecasting System: Implementation and offline validation*. J. Geophys. Res., 118, 1-24, doi:10.1002/jgrd.50488.

[6] Takahashi, T., et al. (2009) *Climatological mean and decadal changes in surface ocean pCO_2 and net sea-air CO_2 flux over the global oceans.* Deep-Sea Res. II, 56, 554-577.

[7] Bergamaschi, P., et al. (2007) *Satellite chartography of atmospheric methane from SCIAMACHY on board ENVISAT: 2. Evaluation based on inverse model simulations*. J. Geophys. Res., 112, D02304, 10.1029/2006JD007268.

[8] Bergamaschi, P., et al. (2009) *Inverse modeling of global and regional CH4 emissions using SCIAMACHY satellite retrievals*, J. Geophys. Res., 114, 10.1029/2009JD012287.

[9] Boucher, O., M. Pham, and C. Venkataraman, (2002) *Simulation of the atmospheric sulfur cycle in the LMD GCM: Model description, model evaluation, and global and European budgets*, Note 23, 26 pp., Inst. Pierre-Simon Laplace, Paris, France.

[10] Guelle, W., M. Schulz, Y. Balkanski, and F. Dentener, (2001) *Influence of the source formulation on modeling the atmospheric global distribution of the sea salt aerosol*. J. Geophys. Res., 106, 27,509– 27,524.

[11] Tompkins, A. M. (2005) *A revised cloud scheme to reduce the sensitivity to vertical resolution*. Tech. Memo. 0599, 25 pp., ECMWF, Reading, UK

[12] Yarwood, G., Rao, S., Yocke, M., and Whitten, G.Z. (2005) *Updates to the carbon bond chemical mechanism: CB05, Report to the U.S. Environmental Protection Agency*, RT-04-00675, Yocke and Company, Novato, California, United States

[13] Hertel, O., R. Berkowicz, and J. Christensen, (1993) *Test of two numerical schemes for use in atmospheric transport-chemistry models*. Atmos. Environ., 27A,16, 2591-2611

[14] Jacob, D.J. H. Liu, C. Mari, and R.M. Yantosca (2000) *Harvard wet deposition scheme for GMI*, Harvard University Atmospheric Chemistry Modeling Group.

[15] Janssen, P.A.E.M. (2004), *The interaction of ocean waves and wind*, Cambridge University Press