Technical Memo

ECMWF
European Centre for Medium-Range Weather Forecasts

857

The ECMWF Scalability Programme: Progress and Plans

Peter Bauer, Tiago Quintino, Nils Wedi, Antonino Bonanni, Marcin Chrust, Willem Deconinck, Michail Diamantakis, Peter Düben, Stephen English, Johannes Flemming, Paddy Gillies, Ioan Hadade, James Hawkes, Mike Hawkins, Olivier Iffrig, Christian Kühnlein, Michael Lange, Peter Lean, Pedro Maciel, Olivier Marsden, Andreas Müller, Sami Saarinen, Domokos Sarmany, Michael Sleigh, Simon Smart, Piotr Smolarkiewicz, Daniel Thiemert, Giovanni Tumolo, Christian Weihrauch, Cristiano Zanna

February 2020

Series: ECMWF Technical Memoranda

A full list of ECMWF Publications can be found on our website under:
http://www.ecmwf.int/en/publications

Contact: library@ecmwf.int

Abstract

The efficiency of the forecasting system on future high-performance computing and data handling systems is considered one of the key challenges for implementing ECMWF's ambitious strategy. This was already recognised by ECMWF in 2013, and has led to the foundation of the Scalability Programme. The programme aims to address this challenge as a concerted action between the Centre and its Member States, but also draws in the computational science expertise available throughout Europe.

This technical memorandum provides an overview of the status of the programme, highlights achievements from the first five years ranging from observational data pre-processing, data assimilation, forecast model design and output data post-processing, and defines the roadmap for the next five years towards a sustainable system that can operate on the expected range of hardware and software technologies.

This point in time is crucial because the programme will have a strong focus on implementation and operational benefit in the next period.

Executive Summary

This paper describes ECMWF's Scalability Programme that was founded in 2013. The programme defines and implements the software infrastructure needed to enable the Centre's forecast production workflow to operate on future high-performance computing and data handling technologies that will be more heterogeneous and require different software methodologies. Only through a concerted software development effort will this technology transfer be possible and allow ECMWF to implement its ambitious strategy. The paper describes both 2014-2019 developments and the 2020-2024 roadmap in detail so that ECMWF and its Member States have available a basis for future coordination and resource planning.

While the first phase of the programme already produced operational benefits, it has mostly been a research phase. The *key achievements* of this first phase are as follows:

1. ECMWF has tested the limits of the existing prediction system on the largest super-computing facilities in the world, and explored the most promising remaining performance optimisation options that can be realised within the existing code infrastructure. For example, mixed precision arithmetics, concurrent execution of model components, overlapping computation and communication, and the use of more efficient CPU[1]-type processors and interconnects are expected to provide code speed-ups of the order of 3 by 2021. No significant further enhancements will be available unless CPU technology improves significantly.

2. ECMWF has established the novel concept of weather and climate dwarfs and benchmarks in the community. This concept allows, (1) to drive performance optimisations targeted at the entire range of new processor technologies in the most efficient way and, (2) to produce realistic estimates of sustained performance of the full system on large-scale, future super-computers. Already now, throughput speed-ups by a factor of more than 20 have been achieved for the most costly model components on single GPUs[2], and a factor of 24 on over 11,500 GPUs, when compared to CPUs. The first ever implementation of a forecast model component on an FPGA[3] resulted in a factor of 2.5 improvement of time to solution and at least a factor of 10 in energy to solution compared to CPUs.

3. ECMWF has built its entire performance enhancement strategy on co-design, namely the co-design of numerical methods and algorithms with code implementation, so that, (1) scientific and computational performance can be traded off against one another if needed, and (2) extensions to coupled modelling and assimilation and atmospheric composition will be future proof. This was possible by creating the flexible OOPS[4] data assimilation system, two dynamical cores operating on the same grid, and the generic *Atlas* data structure framework, all of which have reached a very high level of maturity.

4. ECMWF has undertaken the modernisation of various data handling components of its forecasting system. These include (1) the extension of the IFS[5] I/O[6] server to the wave model and ensemble, (2) the development of a new object-store for model output, supporting novel I/O architectures, and (3) research into novel methods of post-processing leading to critical insight into the successful development of the new product generation software. All these developments have already

[1] Central Processing Unit
[2] Graphics Processing Unit
[3] Field Programmable Gate Array
[4] Object Oriented Prediction System
[5] Integrated Forecasting System
[6] Input/Output

been successfully deployed into time-critical operations, the combination of which has helped to achieve a 5-fold speed-up in product generation, whilst also increasing system robustness. The new product generation software has already been tested with a novel, non-volatile memory extension technology providing a factor 10 speed-up.

5. ECMWF has developed the *Kronos* workflow and benchmark generator, that has been used already in the current HPC[7] procurement providing, for the first time, a full workflow benchmark including model output and product generation, thus being much closer to the reality of our operational system.

6. ECMWF has founded a European collaboration network and established itself as the leading centre performing cutting-edge research at the interface between computational and weather and climate science. This has been achieved through leadership of and participation in several European research projects, providing key contributions to European infrastructure investments provided by the European Commission and to the planning of funding programmes. Through this, the Scalability Programme has complemented the ECMWF core staff investment by an average of €1.5 million per year between 2015 and 2021 with external funds.

7. ECMWF has created a new intellectual pool for scientific computing at the Centre that complements the existing weather and climate research as well as operational expertise, and this ensures that the future forecasting infrastructure will be sustainable and provides the best return on investment for ECMWF's Member States.

The *roadmap* of the second phase, the implementation phase, aims:

1. To fully implement the efficiency gain factor 3 established for mixed precision, model component concurrency, and the OOPS framework with continuous data assimilation on the new HPC infrastructure in Bologna.

2. To focus on two big themes for software development and implementation, namely data-centric workflows and performance portability for both the Earth-system model and data assimilation system. The former addresses how data is being handled along the processing chain between reception and pre-processing, output generation and post-processing, dissemination and archiving. The latter addresses how numerical methods and algorithms can be optimally configured and operated on a range of future computing technology. Both streams combine flexibility and performance with future scientific choices and technology solutions.

3. The first major milestones for data-centric workflows will include (1) the development of an object-based datastore for centralising access to observations, (2) the development of the first version of a data-multiplexing I/O server, also supporting ocean model output, (3) a notification system for model data availability, and (4) a first prototype of a servicing model output to data analytic platforms, such as the European Weather Cloud.

4. The first major milestone for performance portability is the ability to run the IFS-ST[8] at 5 km-scale on a combination of CPUs and GPUs and a fast interconnect on the ECMWF super-computer to be procured in 2024 with at least a factor of 5 better throughput than on today's CPUs. This performance requires the most costly parts of the IFS model and OOPS data assimilation to be adapted to GPUs and relies on a full implementation of the *Atlas* data structure handling framework. This is the lower-effort, less intrusive option for ensuring benchmark readiness for the next procurement.

[7] High-Performance Computing
[8] Spectral-transform IFS

5. The acceleration by a factor of 5 strictly refers to time-to-solution, but implies even more reduced energy-to-solution cost of the forecast model when compared to today due to the use of more energy efficient processor technology.

6. The second major milestone for data-centric workflows will include (1) a prototype for the ingestion of IoT[9] observations and the integration in the observation data store, (2) the generalised usage of the new I/O server supporting IFS-FVM[10], COMPO[11] and on-the-fly post-processing, (3) to feature a complete version of a DaaS [12] cloud infrastructure and full access to IFS model output as data hypercubes, supporting efficient access for HPDA[13] and ML[14] algorithms.

7. The second major milestone for performance portability is to ensure that the Earth-system forecasting system will be fully open to upcoming technology solutions, thus avoiding vendor lock-in, through the implementation of a DSL[15] tool-chain optimised for stencil-based and single-column operations. The tool-chain will be based in parts on automated code extraction and translation tools as well as the *GridTools/Atlas* libraries providing processor hardware dependent back-ends optimised for time and energy to solution. This is the higher-cost, more intrusive option that demonstrates the benefit of the DSL-concept and delivers a template for the future.

8. Another major milestone is the ingestion of ML tools and techniques to optimise and accelerate observational data selection and quality control, provide performance speed-ups through surrogate model components, and enhance model output processing and product generation through data analytics. If the entire model physics parametrizations could be replaced with neural networks, a speed-up by 20% could be expected.

The Scalability Programme has clearly established ECMWF's vision and leading role in this area, but the programme's success is also more than ever relying on collaboration with Member State organisations, HPC centres, vendors, and academia, also beyond European boundaries. The complexity of the task requires concerted developments and funding. Both coordination and funding at significant scale have been achieved so far, but ECMWF and our community need to ensure that both are sustained throughout the important implementation phase of the programme in the next five years.

Efficiency vs Scalability
Efficient code execution is the strongest driver for performance enhancement, and scalability contributes to efficiency as improving scalability allows to shorten code execution time through parallelisation. The main performance metric for code execution is energy to solution in units of kWh. To optimise energy to solution requires to also reduce data movement that consumes about 10 times more energy than calculations, and to use low power processors.

[9] Internet-of-Things
[10] Finite-Volume Module
[11] Amospheric composition version of the IFS
[12] Data-as-a-Service
[13] High-Performance Data Analytics
[14] Machine-Learning
[15] Domain Specific Language

1 Introduction

1.1 Background

In 1965, Gordon Moore, a co-founder of Intel Corporation, observed that the number of components (i.e. transistors) per integrated circuit doubles every year on the back of continuous advancements in manufacturing processes [78]. This observation, although revised a decade later to every two years, is commonly known as Moore's law.

The semiconductor industry has kept these predictions alive over the past decades as seen in Fig. 1 and converted the exponential growth in the number of additional transistors into optimisations and architectural features which made processors significantly faster and more efficient than their predecessors.

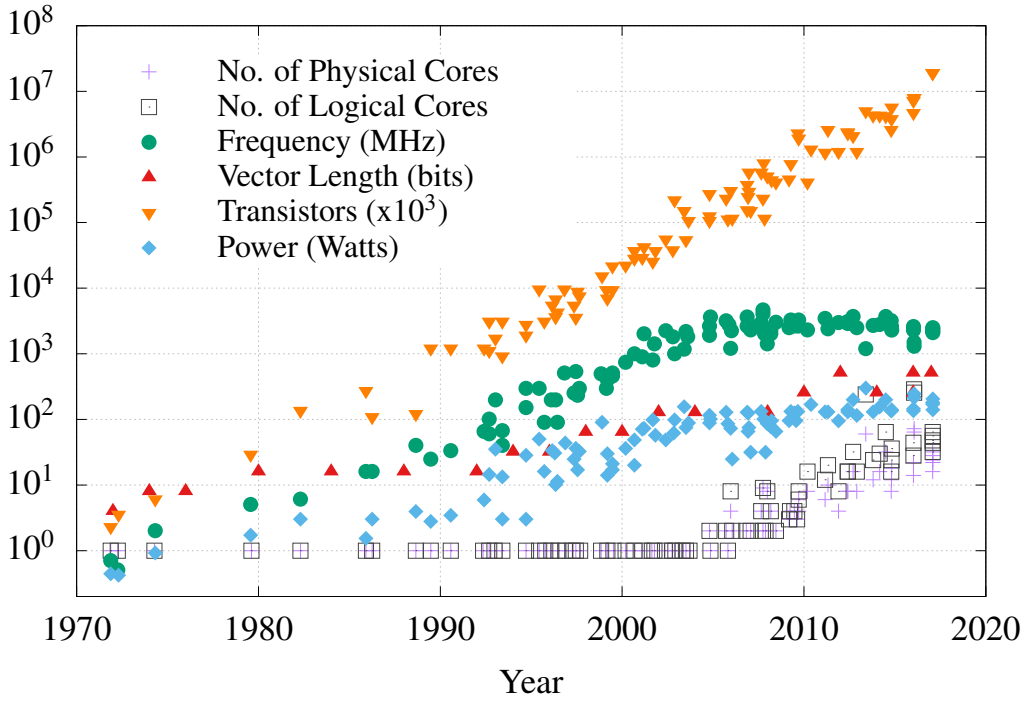


Figure 1: Trends in microprocessors over the past 46 years. Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten. Data between 2010-2015 collected by K. Rupp. Latest data between 2015-2019 collected and plotted by the authors.

For many years, Moore's law was accompanied by Dennard scaling [27]. Robert Dennard demonstrated in 1974 the proportional relationship between the supply of voltage and current and the linear dimensions of a transistor. In essence, as the size of a transistor shrunk, so did the required voltage and current. This allowed for circuits to operate at higher frequencies for the same power and thus led to the increases in clock frequencies that we see in Fig. 1. This trend continued until 2004 and reached its peak in the 1990s when clock rates would double every 18 months.

The combination of extra transistors as a consequence of Moore's law coupled with the continuous increase in clock frequencies at which they operated allowed CPU designers to double the performance of processors with every new generation. These improvements would automatically translate into faster

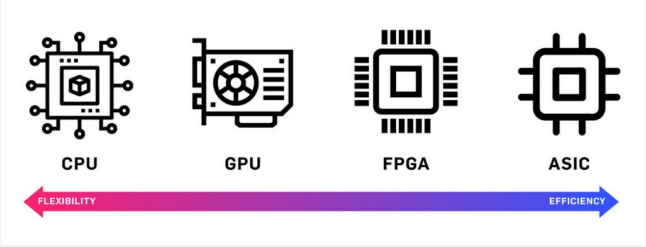
execution of applications in return for little or no changes to the programming paradigms.

However, as transistors shrunk to smaller and smaller scales, it became increasingly challenging to offset current leakage and to dissipate heat efficiently without affecting the integrity of the device. As a result, increases in clock frequencies were deemed to be no longer profitable from circa 2004 onwards as seen in Fig. 1.

The end of frequency scaling has forced the semiconductor industry into a paradigm shift and has led to the advent of multicore processors. Instead of using the additional transistors as a result of Moore's law to build a single monolithic processor, they were used to replicate and integrate multiple cores running at a lower frequency on the same die. The advantages of this approach are based on the fact that reducing the clock frequency of a single core by 20% results in 50% less power consumption at a cost of 13% drop in performance [91]. Thus, if work is divided equally between two processing cores operating at 80% of the frequency of a single monolithic processor, throughput is increased by 73% in return for the same power usage. As a consequence, hardware designers started integrating more and more physical cores on the same die, a trend that continues from 2004 onwards as seen in Fig. 1 where today's multicore CPUs can contain as many as 64 cores.

However, the main drawback of multicore processors is that translating this theoretical increase in performance and energy efficiency into palpable application acceleration mandates the explicit exploitation of parallelism. Whereas the previous paradigm of increasing clock frequencies and improving serial performance required modest programming efforts, the multicore trend only favours parallel applications to the detriment of serial ones. As a result, the burden is placed on the shoulders of application developers to revisit and rethink their existing implementations so as to fully exploit the available parallelism as more and more cores are integrated onto a die [85].

The range of processor types



The hardware of future generations of supercomputers will become more heterogeneous with CPU [a] and GPU [b], various instruction sets (RISC [c], CISC [d], etc.), programmable hardware such as FPGA [e], or even customised hardware for specific applications such as ASIC [f], for example hardware that is dedicated to machine learning such as Google's TPU [g].

[a] Central Processing Units
[b] Graphics Processing Units
[c] Reduced Instruction Set Computer
[d] Complex Instruction Set Computer
[e] Field Programmable Gate Arrays
[f] Application-Specific Integrated Circuits
[g] Tensor Processing Units

This burden is further exacerbated by the emergence of manycore processors such as graphical processing units (GPUs). These architectures push the boundaries of the multicore approach even further by integrating tens of hundreds of cores on the same die. This is made possible by reducing clock rates even further and by discarding architectural features geared towards serial throughput which require a significant amount of on-die logic [86] such as out-of-order execution or complex branch predictors in favour of more space and energy efficient features such as wide vector units or large number of threads.

Although the "slower" core in a manycore architecture will operate at a lower performance compared to

the more powerful core in multicore processors, the total compute throughput of the manycore processor when all of them and their features are exploited in unison will be higher than that of the multicore system [14]. However, for this to be true in practical terms depends on whether the application can not only scale linearly with the number of cores on the device but also across all other existing levels of parallelism (i.e. thread, data and instruction). Otherwise, applications will see significantly worse performance on manycore devices than on multicore systems due to the slower cores that lack features oriented towards single-thread performance. The one saving grace is that multicore and manycore processors share much of the same architectural features. As a result, optimisations for one architecture will most likely be of benefit to the other.

Thus, one returns to the issue beforehand where making effective use of both multicore and manycore processors requires the exploitation of parallelism across different levels of granularity. Understanding what these levels are and ways through which one can exploit them will see an application perform well in the era of multicore and manycore architectures which, judging by the trends in Fig. 1, is here to stay.

Since achieving high performance on both multicore and manycore processors mandates the explicit exploitation of parallelism across different levels, the maximum speed-up that an application can exhibit will be limited by Amdahl's law [56, 14].

Amdahl's law [3] states that the speed-up that can be attained by an application that is executed in parallel is limited by the percentage of execution that is performed serially such that:

$$S(N) = \frac{1}{(1-P) + \frac{P}{N}} \tag{1}$$

where $S(N)$ is the speed-up as a function of N processors, P the fraction of code that is executed in parallel and N the number of processors this is parallelised on. Therefore, as $N \to \infty$, $S(N) \to \frac{1}{(1-P)}$

In practical terms, the limitations imposed by Amdahl's law are that the maximum speed-up that is attainable for an application that spends 25% of its overall runtime in sequential execution is a factor of four, irrespective of the number of processors (i.e $\frac{1}{(1-0.75)} = 4$).

Consequently, applications that will perform best on multicore and manycore processors are those that can maximise P as well as make good use of the underlying architectural features. Furthermore, the limitations in speed-up imposed by Amdahl's law are applicable across all of the available levels of parallelism in modern processors where not exploiting one granularity such as data parallelism leads to a drastic limitation of the maximum attainable performance.

A trend that is not featured in Fig. 1, although as important, is that of the growing disparity between the performance of the processor and that of the memory system. [112] observed in 1995 that although both microprocessor speeds as well as memory bandwidth grew exponentially, they did so at different exponents therefore the difference between them also grew at an exponential rate. The authors coined this trend as "the memory wall" and argued that if this were to continue at the same pace, it would limit the performance of almost all applications to that of the memory system, thus making future advancements in microprocessors redundant. The emergence of multicore and manycore processors have exacerbated this problem. As the number of cores per processor increased, so did floating point performance as arithmetic units were replicated as well. In contrast, other components such as the number of memory channels connecting the memory system to the processor did not scale at a similar pace [79]. Consequently, although peak floating point performance almost doubled with every processor generation, the ability of the memory system to keep all cores and their arithmetic units busy with data lagged further and further behind.

This is evidenced in Fig. 2 which presents a comparison between peak floating point performance and peak memory bandwidth of high-end Intel multicore CPUs, NVIDIA Tesla GPUs and Intel Xeon Phi processors released between 2007 and 2017. On manycore architectures such as NVIDIA Tesla GPUs or Intel Xeon Phi processors, this imbalance is alleviated due to the implementation of on-package high bandwidth memory such as the MCDRAM in the Intel Xeon Phi Knights Landing architecture and HBM2 and HBM3 in the NVIDIA Tesla P100 and V100 GPUs based on the Pascal and Volta architectures.

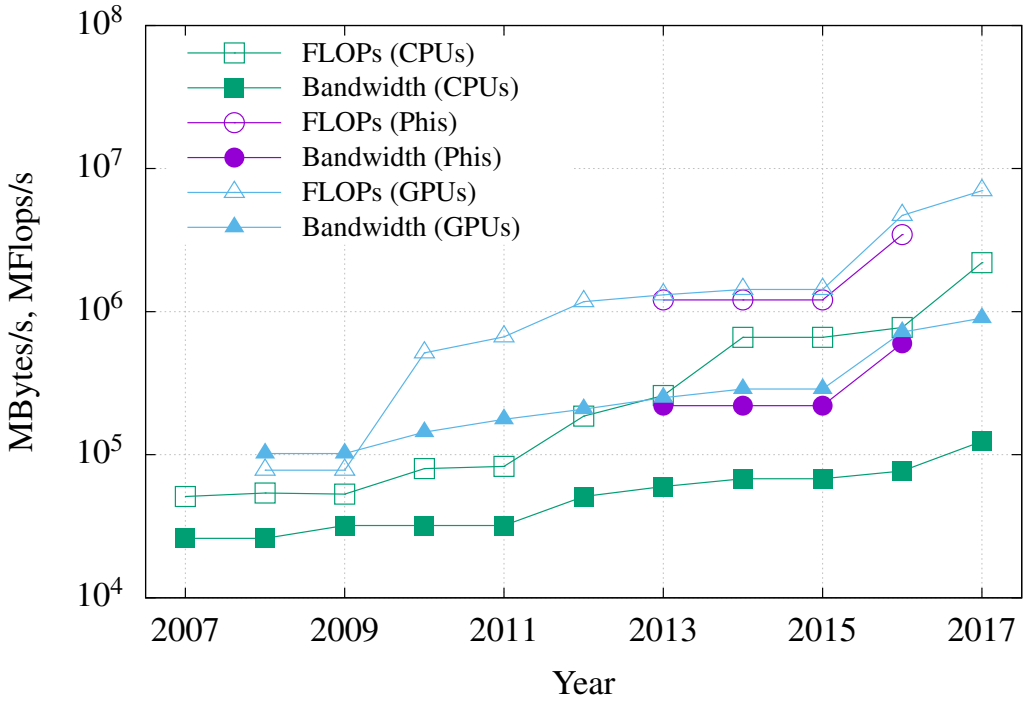


Figure 2: Comparison between peak floating point performance in double precision and peak memory bandwidth across Intel Xeon multicore CPUs (CPUs), NVIDIA Tesla GPUs (GPUs) and Intel Xeon Phi processors (Phis) spanning the last decade. Data courtesy of [92] with minor updates by the authors.

A consequence of the growing disparity in performance between the processor execution speed and the bandwidth and latency of main memory is that the number of floating point operations required to keep all of the available arithmetic units busy for every byte of data retrieved from main memory (i.e., arithmetic intensity) has increased considerably. This is highlighted in Fig. 3. While this ratio is dependent on the machine balance and varies across architectures, it is still growing on the multicore CPUs as previously seen as well as on the NVIDIA GPUs, albeit at a different rate. The implication that this has on application performance is that algorithms that were previously bound by the computational capability of the processor, such as dense matrix-vector operations, are now limited by the performance of the memory system. As a result, techniques for improving the exploitation of the cache hierarchy and which limit the exposure of an application to the latency and bandwidth of main memory will have a high impact on modern architectures and especially on multicore processors.

In addition to processor and memory trends, the overall performance of large-scale applications on HPC infrastructures depends on several other hardware components such as the interconnect, memory size per processor, data latencies and bandwidth across the memory hierarchy, I/O subsystems as well as key system software components like the file system - providing access to storage - and job schedulers. Among

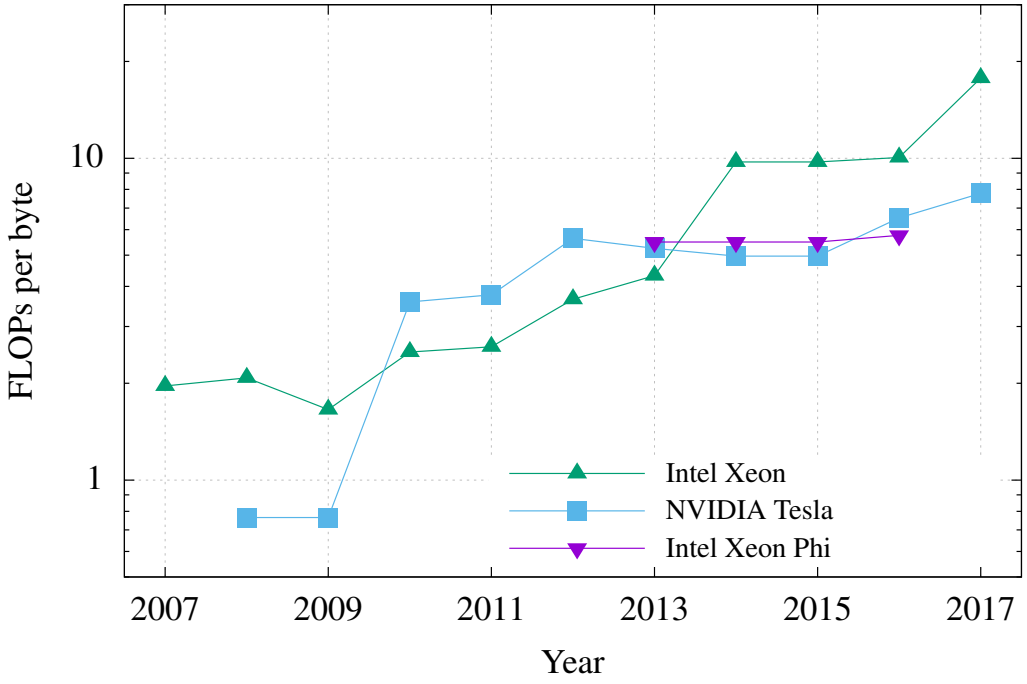


Figure 3: Number of floating point operations in double precision per byte from off-chip memory across Intel Xeon multicore CPUs, NVIDIA Tesla GPUs and Intel Xeon Phi processors spanning the last decade. Data courtesy of [92] with minor updates by the authors.

all of these, the interconnect is the most critical for running large scale applications on increasingly higher number of nodes and has seen significant improvements over recent years with bandwidth injection rates reaching 16 GB/s on ECMWF's Cray XC-40 system. On leading systems such as the DoE[16] IBM P9 'Summit', data rates are up to 100 GB/s, while on future systems such as the DoE's Frontier, these are projected to reach up to 12 TB/s. However, although injection bandwidth has increased significantly between different generations of interconnects, latency has seen very modest improvements. Consequently, although significantly more data can be injected onto the network, the time it takes for this to arrive at its destination has not improved at the same rate. Therefore, a growing importance is placed on the ability of applications to overlap computation and communication in order to fully exploit the capabilities of modern interconnects.

Software programming models have also evolved to suit each generation of CPU. Following the introduction of distributed memory systems from the early vector processing machines, the MPI[17] was adopted as the industry-standard method for communicating between CPUs. With the advent of multi-core processors, OpenMP[18] enabled shared-memory parallel programming. Although shared-memory programming may not be essential for multi-core CPUs, as multiple processes running on the same CPU could instead communicate via message passing using MPI, it has been useful in practice since it can reduce the amount of memory consumed by the application and can allow for a more efficient communication between tasks using shared-memory as a medium. As a result, hybrid parallelism using MPI for communication between nodes, and OpenMP for shared-memory parallelism inside the node, is a widely

[16] Department of Energy
[17] Message Passing Interface
[18] Open Multi-Processing

used programming model at present.

An alternative to MPI is the so called PGAS[19] programming model. These allow global addressing in an implicit fashion across distributed memory systems, while enabling local access semantics. Two common PGAS languages with Fortran support are Co-Array Fortran, part of the Fortran standard since 2008, and GPI[20] developed by the Fraunhofer institute in Germany. One of the advantage of PGAS languages over MPI is that they are based primarily on the concept of one-sided communication where the receiver does not actively take part in the communication. In contrast, although the MPI standard supports one-sided communication as well as of version 2.0, the majority of applications still implement communication in a message passing style where both sender and receiver take part in the communication process and therefore have to synchronize at a given point.

New programming models are also required for making efficient use of accelerators such as GPUs. There are currently several different options available ranging from proprietary, such as NVIDIA's CUDA[21], to ones based on open standards such as OpenCL[22], OpenACC[23] and OpenMP. Best performance is usually obtained with proprietary options (e.g., NVIDIA CUDA) albeit at the cost of locking the implementation to the architecture of a particular vendor while also requiring major levels of re-write of existing code. Although the OpenCL framework mandates extensive code re-writes as well, it targets a wider range of processors and accelerator architectures with a particular focus on performance portability. However, the most popular approach for porting applications to GPUs is via the use of compiler directives based on the OpenACC and OpenMP standards. These programming models require significantly less code intervention than CUDA and OpenCL in return for reasonable performance.

Until recently, problems that could make efficient use of GPUs were also limited to those whose data set, once distributed, could fit onto the limited device memory, and problems that had a small number of easily offloadable, compute intensive tasks. This restriction has been significantly lifted by the arrival of unified memory (and managed memory options of the compiler), which allows the GPU to access host memory transparently, and also by the significant increase in speed and functionality – for example hardware page migration – of the latest host-GPU interfaces.

In the context of ECMWF, Fig. 4 shows the history of sustained floating point operations achieved at ECMWF between 1979 and today together with model resolution upgrades that explain the bulk of the computational cost increase of the model over time, and the size of the tape archive. The latter is explained by the combination of field size changes following resolution upgrades and product diversity enhancements. The present procurement will still benefit from CPU based on Moore's law scaling; however, no future system will follow this trend without an investment in heterogeneous processing, memory hierarchy and network topology architectures.

As these technologies emerge, the most important issue is a software development gap that is caused by the lack of scientific and programming flexibility needed to optimally exploit such technology. This paper presents research and development efforts to fill this gap for ECMWF's weather prediction workflow. Most of the software developments can be generalised for other centres and potentially beyond weather and climate prediction.

[19] Partitioned Global Address Space
[20] Global address space Programming Interface
[21] Compute Unified Device Architecture
[22] Open Computing Language
[23] Open Accelerator programming model

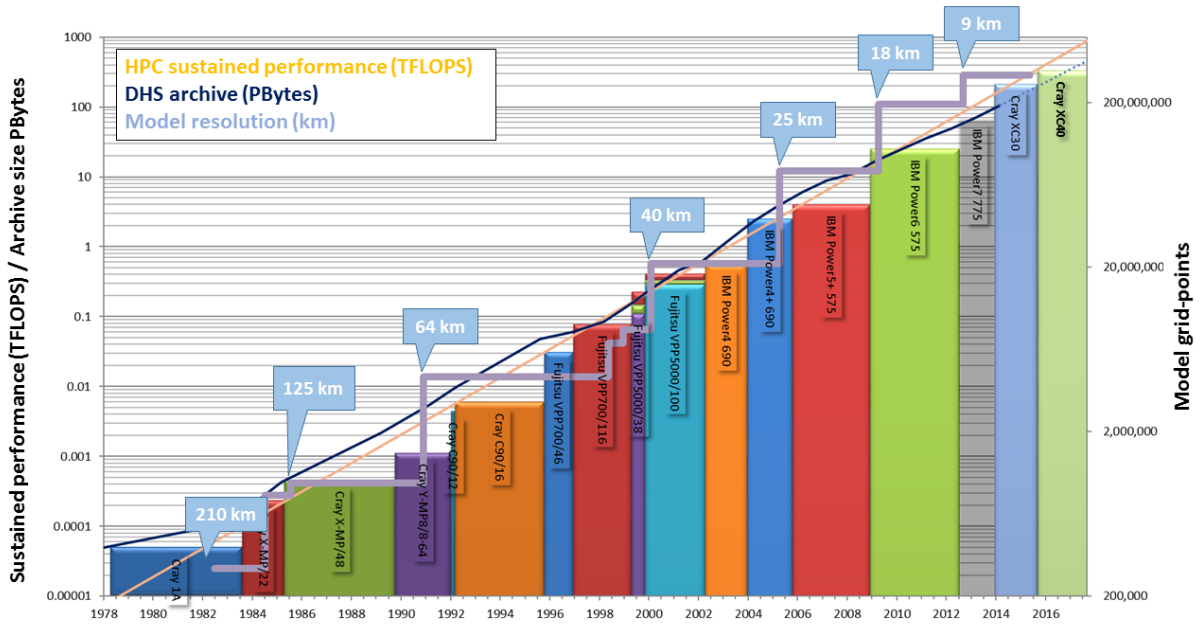


Figure 4: Historic growth of sustained HPC performance (in 10^9 FLOP/s), archived data volume (in 10^{12} bytes) and model grid-points following spatial resolution upgrades.

1.2 Present performance

The IFS software development is a long-lasting endeavour to achieve world-class medium-range numerical weather prediction. The development of the code, shared between Météo-France, ECMWF and various limited-area modelling communities, has been ongoing since the 1980s. It is written largely in Fortran of various standards (F77, F90, F2003/08) and some C.

The code's design reflects its long history, and the original design goal of integration is denoted by the name. The main IFS binary executable is used for most of the tasks in the forecasting workflow, and the control layer code selects between the numerous tasks that the software suite can carry out.

The IFS has a hybrid MPI/OpenMP parallel implementation. MPI is used to partition the grid-point domain (i.e. spatial domain decomposition) and similarly, the computations in spectral space are also partitioned into different domains. Work inside each domain is shared among cores on a socket thanks to very coarse level OpenMP work-sharing loops. These work-sharing constructs, along with the design of the bespoke data structures, avoid potential performance issues such as false sharing. Fields data structures, holding the atmospheric state, are laid out in memory in blocks of vertical columns. The block structure, with NPROMA columns per block, was originally designed for vector machines, but has proved over the years to be well suited for tuning computations on more general architectures.

1.2.1 Operational suite configuration

To give context to the remaining discussions, a brief overview of the IFS system and ECMWF's operational forecast suites is given here. The main operational analysis and forecast suites at ECMWF consist

of the following components:

- Ensemble of data assimilations (EDA);
- High-resolution single analyses and forecasts (4DV and HRES);
- Medium-range (leg-A) and monthly (leg-B) ensemble forecasts (ENS);
- Seasonal ensemble forecasts (SEAS);
- Ensemble reforecasts (REF);
- Ocean ensemble of analyses (OCEAN);
- CAMS[24] atmospheric composition single analyses and forecasts (COMPO).

COMPO has been added to gauge the relative cost of atmospheric composition modelling, relevant for future model upgrades. The most relevant information on suite configuration and HPC allocations valid for the present cycle (46r1) configuration is summarised in Tab. 1; see also [17].

For the last 15 years, ECMWF has operated two separate HPC clusters to enhance resilience and maintenance: if one fails there is a back-up, and routine maintenance (such as system patches) can be applied on one cluster at a time without affecting operations. This philosophy may change in the future and more of the available overall node count may be allocated to operations in the critical path.

Note that every day during the production hours, the operational suite uses up to a peak of about 85% of the usable nodes of one cluster; this safety factor is imposed for technical reasons.

In addition to the components listed, the 06 UTC and 18 UTC boundary-condition runs, which are not shown, represent a substantial use of compute resources. Furthermore, there are many smaller tasks in the production suites that are individually too minor to list explicitly, such as observations processing in the analysis-type suites, EDA error calculation, stand-alone wave model runs, singular-vector calculations in the ENS suite, and other general post-processing tasks. These also represent a substantial additional use of computer resources.

Table 1: IFS analysis and forecast suite configuration and computing allocations valid for cycle 46r1 (Explanatory notes: Members = ensemble size; Resolution = outer loops for assimilation; Length = assimilation window or forecast range; Nodes = Cray XC-40 compute node allocation; Time = average wall-clock time; Relative cost = of operational allocation. Starred values of relative cost have been estimated as a daily average of the total usage over the previous 365 days). Note that all suites are run twice per day, except ENS leg B and REF (twice per week), COMPO (once per day), SEAS (once per month, 13-month extension once every 3 months). All HRES, ENS and COMPO are coupled with land, wave and ocean models, all EDA and 4DV are coupled with land and wave model. The OCEAN suite has been excluded as up-to-date information is not available.

Suite	Description	Members	Resolution	Length	Nodes	Time (min)	Relative cost
4DV ED	Early delivery 4D-Var analysis	1	9 km L137	6 hours	352	59	4.9%
SEKF	Surface analysis for 4DV EC	1	9 km L5	6 hours	352	3	0.2%

continued on next page...

[24] Copernicus Atmospheric Monitoring Service

...continued from previous page

Suite	Description	Members	Resolution	Length	Nodes	Time (min)	Relative cost
HRES	High-resolution forecast	1	9 km L137	10 days	352	60	4.9%
HRES PGEN	HRES product generation	1	9 km L137	N/A	300	62	4.3%
4DV LWDA	Long-window 4D-Var analysis	1	9 km L137	12 hours	352	63	5.2%
SEKF LWDA	Surface analysis for 4DV LWDA	1	9 km L5	12 hours	352	5	0.4%
HRES LWDA	High-resolution forecast (short)	1	9 km L137	60 hours	352	15	1.2%
EDA	Ensemble of data assimilation	51	18 km L137	12 hours	1428	54	18%
EDA FC	EDA forecast (short)	51	18 km L137	15 hours	1428	9	3%
EDA JB	EDA background errors	51	144 km L137	N/A	196	25	1.1%
ENS leg-A	Medium-range ensemble forecasts	51	18 km L91	15 days	1530	82	29.3%
ENS leg-B	Monthly ensemble forecasts	11	36 km L91	46 days	765	55	1.4%
ENS PGEN	ENS product generation	51	18 km L91	N/A	168	48	1.9%
REF	Ensemble reforecasts	51	18 km L91	46 days	220	variable	23.5%
SEAS 7	Seasonal forecasts (7 months)	51	36 km L91	7 months	408	variable	0%
SEAS 13	Seasonal forecasts (13 months)	15	36 km L91	13 months	120	variable	0%
COMPO	CAMS 4D-Var analysis	1	40 km L137	12 hours	42	58	N/A
COMPO	CAMS forecast	1	40 km L137	5 days	63	25	N/A

Even with these omissions, the production runs are very costly, and represent nearly 20% of the total node-hours available from a single cluster. The e-suites run in both research and pre-operational trials add significantly to this cost.

Moreover, because of the highly peaked nature of the production runs, the instantaneous usage is often much higher than the average of 20%. During the so-called 'model hour', when both the ENS forecasts and the HRES forecasts are running, along with their product generation suites, about 85% of the usable nodes of one cluster are in use at peak-time. This is the reason why the current resolution is the highest

possible today, given that the operational production must be completed in the same amount of time every day even in case the operational cluster has problems, and clusters need to be swapped. This also means that testing IFS cycle upgrades at the full operational resolution must be planned carefully, and can only be done with very few configurations.

Integrated over all suites and applications deployed on the HPC system, at least three types of workload need to be distinguished:

- *Time-critical* work is characterised by the need to finish in a fixed, short duration and at the same time every day, week or month. ECMWF's operational analysis and forecast runs clearly fall into this category. As the amount of work is increased in each run over time (by increasing resolution and reducing timestep, complexity, ensembles) the time-to-solution remains the same, making this a strong scaling problem where weak scaling properties are also essential.
- *Capability* work pushes the limit of a machine and application to achieve scientific breakthroughs. Usually, it is characterised by the need to use the largest possible resource allocation, to achieve, for example, the highest resolution possible. Time to solution is not necessarily a strong constraint, since this work is in most cases of research nature. Because of the focus on ensembles, ENS production runs are arguably not classed as capability work, each ensemble member being only modestly parallel. The largest single node allocation is to the HRES suite, of 352 nodes. However, the sum of the parallel ENS member allocation is the largest capability allocation of any suite.
- *Capacity* or throughput work forms the remainder, and comprises only small parallel tasks that tax neither the machine nor the application software. In ECMWF's case, also a very large number of serial tasks are run at all times. In general, the total machine throughput of such calculations benefits from them being allocated to the smallest number of nodes possible to provide the required memory (unless the application scaling is perfect). The vast majority of research work at ECMWF is run at a low resolution and uses at most a few tens of nodes, hence it falls into this category.

If research work were a continuous, ongoing submission of jobs, then it would be purely a throughput problem. However, we marshal the work into phases which are time-limited, which gives it a time-to-solution characteristic too (time to solution here includes queuing time). The main phasing comes from the creation of IFS cycles. Within a cycle we also have a phased development process which provides a finer granularity.

1.2.2 Cost break-down and scaling

Fig. 5 breaks the overall execution time down into the main IFS model components for the HRES and ENS model components for medium and monthly range on ECMWF's Cray XC-40. While the overall cost of the suites reduces as resolution decreases, the relative cost of model physics and, in particular, NEMO increase drastically with reduced resolution, mostly because NEMO is run at 1/4 degree resolution and with 75 layers in all configurations. For low-resolution research experiments a reduced NEMO configuration is used.

By evaluating the time each component of the IFS uses on different numbers of nodes their scalability can be derived. Figure 6 shows the times in each component for the operational configuration of the IFS using different node counts. Here we see that the grid-point physics show near-perfect scaling as the node count increases, this is possible as the calculation of each grid column can be performed independently, with no communication required. The grid-point dynamics also exhibit good scaling as

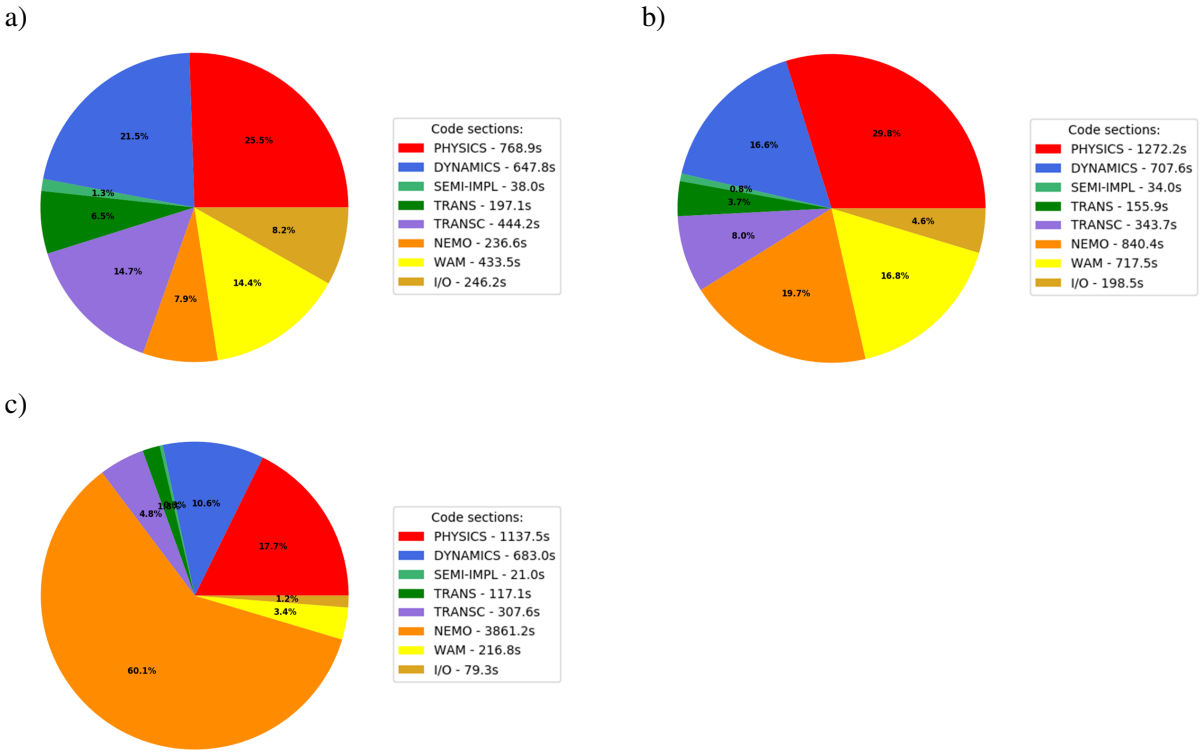


Figure 5: Fraction of forecast run time spent in the different components of the IFS for the TCo1279L137 HRES (a), TCo639L91 ENS (b) and TCo319L91 SEAS (c) forecasts for May 2019.

only 'nearest-neighbour' communication is required. The components related to the spectral transforms (labelled TRANS) and data transpositions (labelled TRANSC) scale less well due to the large amount of communications required. NEMO shows particularly poor scaling in part due to the implicit solver requiring a global communication at each iteration, and due to the inefficient exploitation of shared memory parallelism based on OpenMP in NEMO compared to the IFS. The I/O cost also does not scale, with the execution time increasing for more nodes.

Figure 7 illustrates the scaling behaviour and use of energy with the 9-km TCo1279 forecast on the Cray XC-40. All runs on more than 100 nodes are relevant for time-critical applications and achieve more than 240 forecast days per day.

1.2.3 IFS benchmark runs on external machines

ECMWF has a history of trialling the IFS on external HPC systems beyond procurement benchmarking. These tests help gauge efficiency and scalability of the code on larger allocations and different types of CPU and networks, and provides information on compiler performance. Early examples reach back to runs of a few time steps on an 8-processor Cray-YMP performed at a Cray symposium in 1988. The Cray YMP had a theoretical peak for the entire system of 2.75 Gflops/s, and the IFS broke the gigaflops barrier soon after. The teraflops barrier was broken with the IFS in 2004 on 2,048 IBM p690+ processors using one entire cluster of ECMWF's computer at the time.

A more recent example is ECMWF's trials on an allocation on the Titan Cray XK7-system in Oak

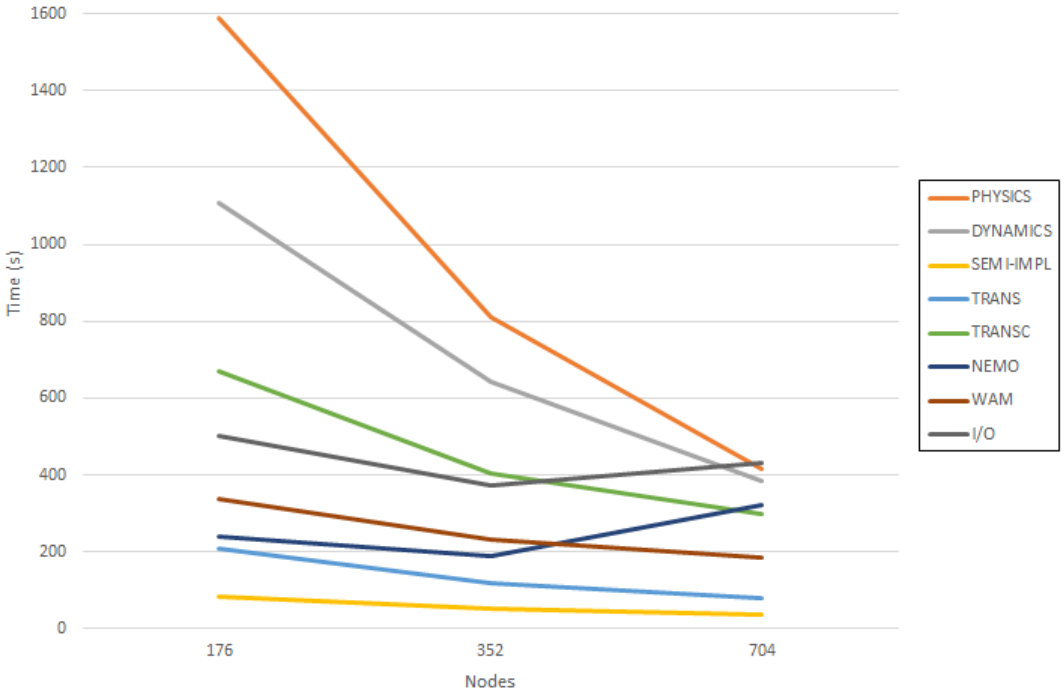


Figure 6: Scaling of each component of an IFS operational forecast on the Cray XC40.

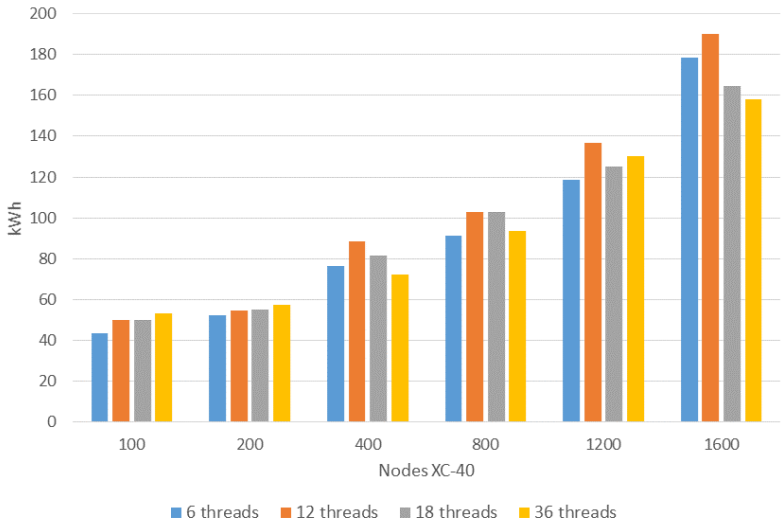


Figure 7: Energy use of the IFS atmosphere-only on the Cray XC40 with different nodes and OpenMP threads in kWh at TCo1279 L137 (9 km), The 12-, 18- and 36-thread configurations used hyper-threading.

Ridge National Laboratory (ORNL) within the CRESTA[25] project. Titan occupied first place on the

[25]Collaborative Research into Exascale Systemware, Tools and Applications

top500.org list in November 2012. On this system, the IFS was scaled up to 220,000 cores, which was not possible with ECMWF's own computer. Already then, experiments were performed with different ways to optimise and overlap computations and data communication achieving good efficiency gains [81].

Mostly stimulated by the requirements from ESiWACE[26] work on estimating the computability of present-day models like IFS and ICON, ECMWF ported the model onto the biggest European supercomputer, the Piz Daint Cray at CSCS[27] in Switzerland. The model resolution was set to 1.45 km, which translates to about 16 billion grid points with the cubic-octahedral grid and 62 vertical levels. The model was scaled across the entire supercomputer up to 4880 compute nodes. This was done without being able to use the GPU processors though, which are integrated on the same nodes as the CPUs of Piz Daint.

On this allocation, the IFS achieved a maximum throughput of 69 forecast days per day (see Fig. 8), which is at least a factor of 5 short of the operational requirement for HRES. When also accounting for the cost of coupling, better vertical resolution and data input and output, this shortfall factor would be rather 240 [96]. This was compared to the GPU capable COSMO[28] model as operated by MeteoSwiss on Piz Daint. On similar terms, COSMO's shortfall factor turned out to be about 110. While the tested IFS performed faster in absolute terms on Piz Daint, COSMO was able to exploit both CPU and GPU located on the same nodes thus making much better use of the heterogeneous architecture of Piz Daint.

Most recently, an opportunity to compare the IFS-ST with the IFS-FVM arose through the US Department of Energy's INCITE programme, which provides access to the Oak Ridge facilities for selected large-scale experiments. A fundamental feature of the spectral-transform model IFS-ST is the need to perform data-rich communication across many tasks twice per time step to move back and forth between spectral and grid-point space. The more tasks are run on more compute nodes, the more communication is involved. Generally, moving data consumes about 10 times more energy than performing the actual calculations [63]. In contrast, IFS-FVM performs predominantly local data communication between neighbouring areas with much less data communication per time step, which leads to improved strong scalability across a wider supercomputer network and potentially more options for resilience.

To test this hypothesis and to exploit accelerators, ECMWF accessed the world's largest machine (Summit IBM-machine, top500.org, June 2019) through the INCITE project. Summit combines 2 Power9 CPUs with 6 NVIDIA Volta GPUs on each of the ca. 4,600 nodes, again in a hybrid configuration, but this time with a well-connected network for better exploiting the massive parallelism of the attached GPUs on each node. Through the INCITE award, ECMWF is pursuing two avenues of investigation, namely to perform a IFS-ST versus IFS-FVM head-to-head comparison on the biggest possible CPU allocation to explore the limits of the spectral-transform model performance, and to port as many IFS components as possible to GPUs and repeat the tests on the hybrid architecture. These simulations will be instrumental for ECMWF's research strategy and will inform the Scalability Programme about key sources of efficiency gains and where to strengthen development efforts.

A first test with the IFS-ST has already been carried out to compare with the Piz Daint runs (see Fig. 8). At 1.45 km, the IFS achieved 112 forecast days per day on Summit, which makes it about two times faster on the same number of nodes. Note that Summit has two Power9 CPUs per node while Piz Daint only has one Intel CPU per node. However, the comparison does not mean that the same energy was used. This is clearly another highly relevant question and one of the reasons why we need to learn how to use the attached GPUs much more effectively so that we can exploit the entire power supplied to the

[26] Centre of Excellence in Simulation of Weather and Climate in Europe
[27] Swiss National Supercomputing Centre
[28] Consortium for Small-scale Modeling

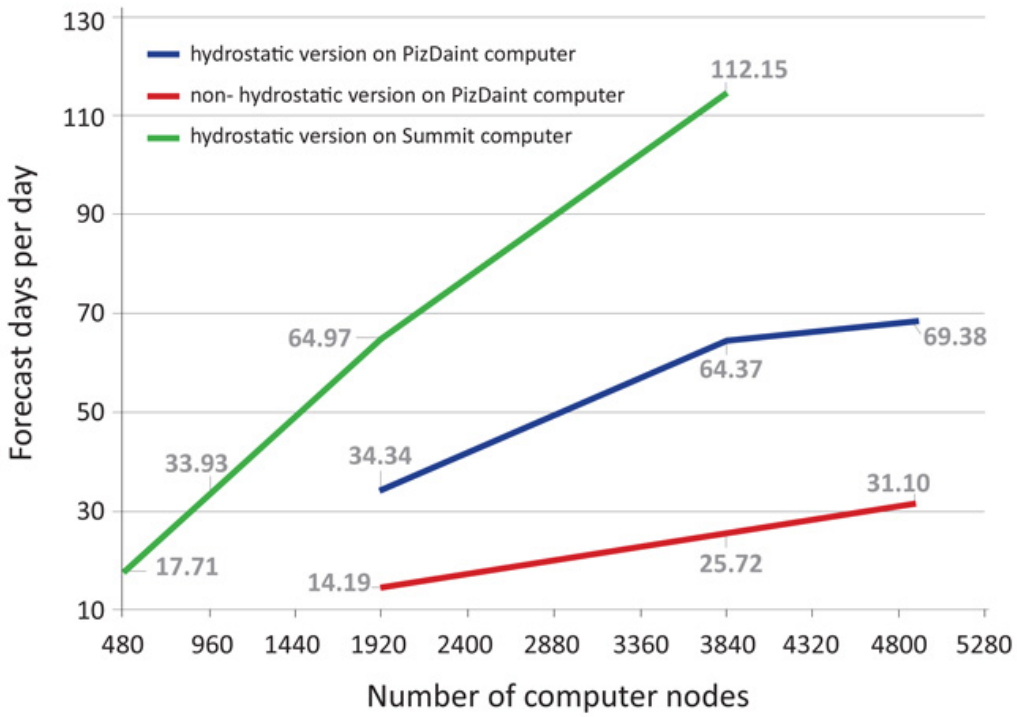


Figure 8: Forecast throughput on Summit and on Piz Daint for the 1.45km (TCo7999), 62-vertical level hydrostatic (H) and non-hydrostatic (NH) IFS using CPUs only with a hybrid MPI/OpenMP parallelisation. The Piz Daint XC-50 Cray has 12 CPU cores per node and Summit IBM nodes have 42 CPU cores.

CPU-GPU node. This assessment still needs to be performed. In support of such an assessment, trials with the spectral transforms on Summit's GPUs have been performed as well (see Section 2).

Measuring speed-up and energy use

This document refers to several speed-up factors and energy measures of the IFS or parts thereof. These have been tested on the available architectures and typically compare against reference simulations on the same architecture, for example measuring the use of CPUs and GPUs for both single and multiple-node allocations on Summit. But they are also compared across systems, for example, Summit vs Piz Daint. These intercomparisons are clearly system dependent as these have different arrangements of processors per node and interconnects.

Energy measurements typically report the energy use on each node, so not using part of a node that has hybrid arrangements of CPUs and GPUs is wasteful. Idle CPUs use at least 50% of the power at full load while GPUs use less than 10%. The present CPU-GPU intercomparisons do not take this into account. The ideal system distributes the workload such that the respective benefits of processors, of on-node and inter-node connections are exploited and idle time is minimised.

The interpretation of speed-up factors and energy measures has to account for these hardware specific dependencies, but they represent a best effort to combine and translate the relative speed-ups into a figure that describes the future acceleration and energy-efficiency potential of IFS compared to the state-of-the-art in 2019.

1.3 ECMWF Scalability Programme

1.3.1 ECMWF's scientific ambition: computing and data handling

ECMWF's 2016-2025 strategy [35] targets the implementation of a global ensemble prediction system at 5 km resolution in 2025 as the cornerstone for achieving its main strategic objectives: namely to produce skilful predictions of high-impact weather up to two weeks ahead, large-scale patterns and regime transitions up to four weeks ahead, and global-scale anomalies up to one year ahead. This target was defined from extrapolating the resolution upgrades from the recent past over the next ten years. ECMWF's new 2021-2030 strategy is being developed now, and it is likely that a similar extrapolation to an even more ambitious target resolution will be made for 2030 In 2017, this ambition was used to make the science and business case for the 2018-2019 procurement of the new HPC infrastructure to be installed in Bologna in 2020. Several options with varying degrees of ambition were defined (see Fig. 9) and ECMWF Council eventually supported the so-called 'enhanced intermediate' option, to ensure that ECMWF will still be able to advance towards its strategic goal while remaining within affordability limits.

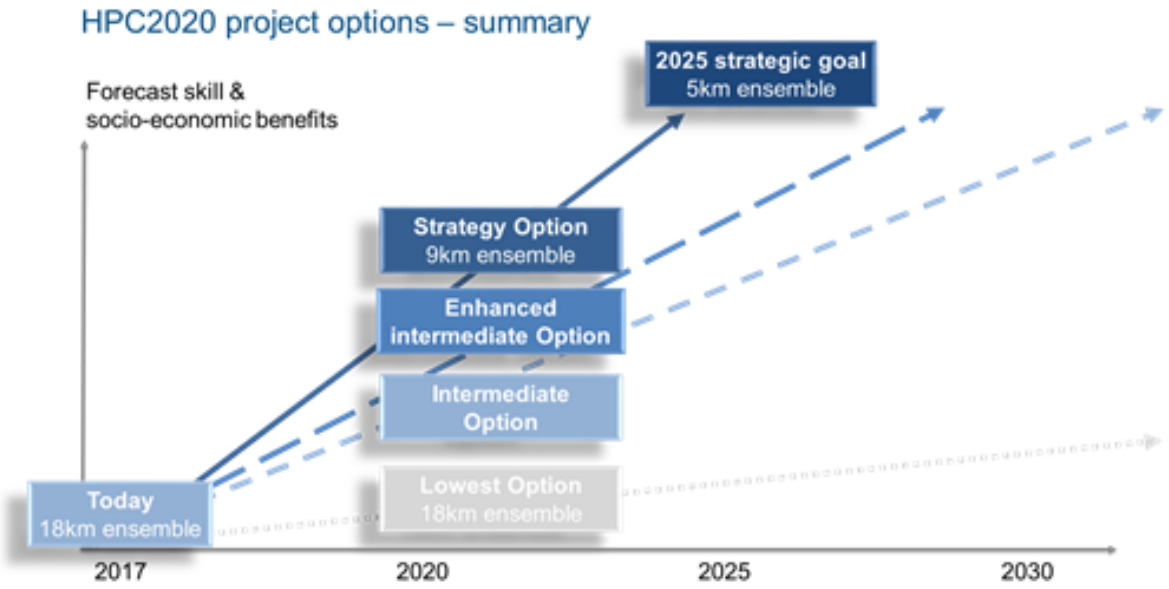


Figure 9: Schematic illustrating the HPC2020 project options and the extent to which they deliver progress towards the ECMWF strategic goals. The Strategy Option permits implementation of a 9 km ensemble and keeps ECMWF on track to deliver its strategic goals. The Intermediate Option allows implementation of a mixed resolution ensemble (or alternatively an intermediate resolution ensemble) and is the minimum required to permit worthwhile progress towards the strategic goals, although full delivery would be delayed significantly. The Enhanced Intermediate Option, while still compromising on the Strategy Option, provides a significant enhancement over the Intermediate Option. The Lowest Option does not permit an increase in ensemble resolution, delivers only very limited progress and puts at high risk ECMWF's world-leading position. Capacity factors for the lowest, intermediate, enhanced intermediate and strategy options were estimated to be 1.5, 2.2, 2.7 and 3.6, respectively (figure and caption taken from HPC2020 Council-91 document).

The performance-requirement estimates focused on EDA and ENS as the most demanding capability and capacity suites, and forecast and outer loop resolution targets of 9 km, respectively, producing overall

cost enhancement factors as shown in Tab. 2. These factors already took into account efficiency gains obtained from near-term workflow and algorithmic changes as well as the expected move to single precision for ENS. The cost estimate was also based on the assumption that all savings can be achieved while maintaining the meteorological performance of the forecasting system. Given the substantial cost of adding atmospheric composition and the unknown benefit for predictive skill in the CAMS-like configuration it was further assumed that the atmospheric composition capability will not be implemented in the full CAMS form and thus not included in the cost estimate.

In terms of capacity computing the impact of the ENS upgrade needs to be propagated into REF, which means that the REF-cost will also increase by a factor of 6.5. The default set-up of research experiments will have to be re-evaluated to ensure that HPC cost and archived data volumes are appropriate, and that lower configuration experiments still provide meaningful information on the entire system's meteorological performance.

Table 2: HPC cost enhancement factor estimates for the strategy option, used in support of the HPC2020 case.

Suite	Upgrade	Cost factor
EDA	Resolution increase in outer loops from TCo639 to TCo1279	3.5
	Resolution increase in inner loops from TL191/TL191/TL255 to TCo191/TCo191/TCo255	1.3
	Addition of one outer loop	1.3
	Ensemble size from 26 to 51	2
	Coupling with ocean (as in CERA)	1.2
	Asymmetric and pre-conditioned EDA	0.5
	Replacement of finite difference with EDA perturbations in SEKF	0.85
	Extraction of 1st screening trajectory from critical path	0.95
	Code improvements	0.95
	Doubling of node allocation on future clusters with 90% scalability efficiency	0.55
	Total	**3.0**
ENS	Resolution increase from TCo639 to TCo1279, and from L91 to L137	12
	Single precision for forecasts	0.6
	Elimination of SKEB model perturbations in ENS	0.95
	Code improvements	0.95
	Doubling of node allocation on future clusters with 90% scalability efficiency	0.55
	Total	**3.6**

The progression to a full sized 5-km ENS in 2025 with appropriate vertical resolution, coupled to a high-resolution ocean will require at least another factor of 15-20 based on the enhanced intermediate option as the HPC system in Bologna will not be sufficient for implementing a 51-member, 9-km coupled ENS. This factor will increase again when projected to the new strategy's target date of 2030. The impact on data handling capability and capacity is very similar.

1.3.2 Organisation of the programme

Already in 2013, the dilemma between the necessary advancement of the ECMWF forecasting system and the lack of business-as-usual progress from the technology roadmaps spawned the need for investing in much more substantial software developments at ECMWF. This motivated the foundation of the

so-called ECMWF Scalability Programme in late 2013. While the term 'scalability' only refers to the parallel-processing contribution to efficiency, the programme actually targeted all aspects of the ECMWF forecast production chain since its initiation. Since then, scalability has been featured as one of the leading future challenges for ECMWF and received much attention in the ECMWF 2016-2025 Strategy and subsequent communications to ECMWF Member States and committees. ECMWF's Council recognised the importance of these challenges and agreed to assign additional resources to the programme from 2015 onwards.

The programme aimed at developing the next-generation forecasting system by addressing the challenges of its extreme-scale high-performance computing and data management needs on future architectures within the expected resource constraints. Key components of model, data assimilation, code adaptation to new and emerging hardware solutions as well as data pre- and post-processing were included in the programme since the beginning to ensure that the efficiency and scalability aspects of the entire forecast production and data dissemination workflow could be tackled.

The programme also took into account that core code components are shared with ARPEGE[29], HIRLAM[30] and ALADIN[31] communities and aimed at an integrated code development approach with active participation of ECMWF Member States. The governance structure of the programme included a board with representation by DWD, the Met Office, Météo-France, MeteoSwiss, and the HIRLAM, ALADIN and COSMO consortia. Apart from creating a body that coordinates computational science with traditional developments at the Centre, the programme's foundation clearly advertised the much enhanced role of computing and data handling constraints to ECMWF staff, but also emphasised the importance of Member State collaboration for being successful.

The main objectives of the programme were to implement a forecasting system that:

- combines flexibility for scientific choices with maximum efficiency achievable on the range of expected future technologies;
- combines resilient workflows for observational input and forecast model output data with maximum efficiency achievable on the range of expected future technologies;
- implements computing and data handling performance metrics and a framework for code testing allowing the quantitative assessment of code efficiency and scalability.

The individual projects of the Scalability Programme's first phase (2014-2019) ingested already existing efforts that have been targeting efficiency gains but also created entirely new efforts (see Fig. 49 in Section 7 in the Appendix).

Apart from the added core resources, the acquisition of third-party funding represents an important target for the programme. Beyond the provision of staff resources, externally funded projects greatly help to (a) establish ECMWF's leading role in applied-computational scientific research in the weather and climate prediction community, (b) forge research collaborations between ECMWF, its Member State organisations, academia and HPC-centres, and (c) strengthen the collaboration with hardware vendors and emphasise the importance of weather and climate prediction requirements in future technology roadmaps.

An important vehicle in this dialogue became ECMWF's membership in the ETP4HPC[32] since 2015 and its contribution to the formulation of their strategic research agenda that builds the foundation for the

[29] Action de Recherche Petite Echelle Grande Echelle
[30] High Resolution Limited Area Model
[31] Aire Limitée Adaptation dynamique Développement InterNational
[32] European Technology Platform for High-Performance Computing

European Commission's FET [33] funding programme. Since 2015, ECMWF has been extremely successful at acquiring European Commission funding, and both resources and collaborations have contributed key developments towards achieving the above objectives. Further, ECMWF has succeeded to raise the importance of future HPC and data concerns at WMO[34] [16, 8].

Section 2 describes these achievements in more detail, from which the main strands of development for the second phase of the programme are derived in Section 4, followed by an implementation roadmap in Section 6.

1.3.3 Efficiency gains

Fig. 10 provides a general idea of system levels, namely workflows, suites and tasks. Examples are the operational end-to-end ensemble prediction or the reanalysis workflow, 10-day forecasts or 4D-Var analysis suites, and model integration or minimisation tasks. Workflows include the entire dependency graph of individual suites and all related computing and data handling operations, while suites are well-defined sub-units of workflows. Tasks are usually single executables and may be limited to either computing or data handling.

This structure also defines the sources of efficiency and where specific hardware components may be targeted - as also shown in Fig. 10 - from node-level for tasks up to full-system-level for suites and workflows. For example, the fact that the sustained performance of the IFS (forecast model) is presently about 4% mostly relates to the parallelism and efficiency achievable on scalar CPU-type architectures with limited memory bandwidth, and is thus a task/node-level problem. On the other hand, the huge data handling overhead of the model output post processing mostly relates to repeated read and write operations from memory to disc, and is thus a suite and workflow/file-system-level problem.

Section 2 will go into more detail about past work targeting workflow, suite and task-level efficiency enhancements exploring the relevant hardware components and software options while Section 4 will define the research foci and implementation paths for the future.

2 The first phase: 2014-2019

The first phase of the Scalability Programme has focused on core aspects of data processing in the workflow, and on developing model & data assimilation algorithms and configuration systems that promise greater flexibility in the future in terms of scientific and performance choices. Section 2.1 describes the data workflow redesign and the tools introducing the data-centric view on workflow efficiency and performance portability including application examples. As for processors, new technologies for filling latency and bandwidth gaps in the memory hierarchy are being explored. Section 4.1 develops this concept towards operational implementation. Section 2.2 presents the development of tools at code level which allow the generic handling of data structures across the IFS and which complement the algorithmic work on the new dynamical core IFS-FVM and the object oriented data assimilation framework OOPS. This section also includes advanced research on new programming models allowing to run code on new processor architectures. Section 2.3 highlights several examples of performance gains within the existing code infrastructure that will be brought into operations in the short term. This will be integrated into a

[33] Future and Emerging Technology for HPC
[34] World Meteorological Organization

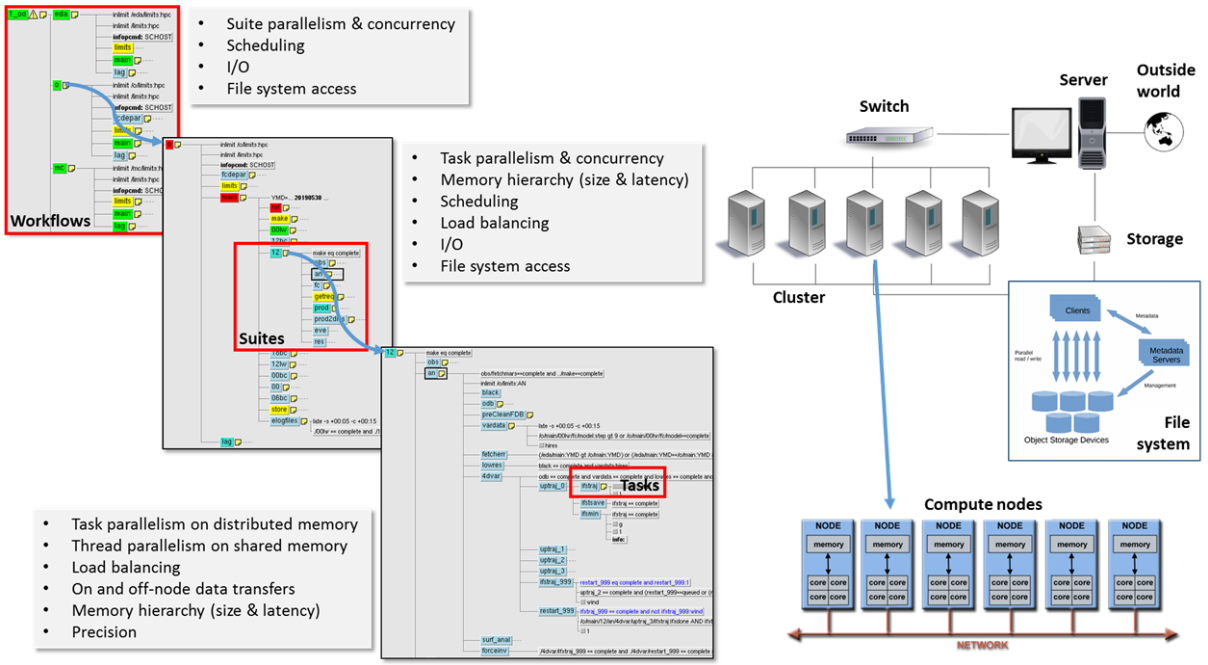


Figure 10: Sources of efficiency gains for workflows, suites and tasks, and hardware components from component to system level for both computing and data handling.

new framework for full algorithmic and hardware flexibility in Section 4.2, to be achieved near the end of the second phase of the Scalability Programme.

2.1 Data processing: observation input and model output

The combined effect of improvements to computing hardware, infrastructure, middleware and scientific software contributes to an exponential increase of ECMWF's capabilities to produce useful output. The investments for higher-resolution analyses and forecasts, increased ensemble size and greater model complexity directly translate into an impressive growth rate of the volume of data that ECMWF ingests, produces, distributes and archives. Fig. 11 shows past and expected growth of data volumes in the forecasting pipeline for (a) observational data used in data assimilation (mostly from satellites), (b) total model output for the ensemble forecast suite and (c) the total amount of data disseminated over the internet per day.

The main drivers for observational input data are hyper-spectral infrared satellite sounders (IASI, IASI-NG, MTG-IRS)[35]. The observation processing pipeline consists of a pre-screening stage, where quality control takes place that is independent of the model background, and can thus be applied before the execution of the first trajectory which produces the so-called innovations. Today the observation pre-screening processes over 20 billion observations in every 12 hour assimilation window. The remaining 400 million observations are then fed into the IFS where background-dependent quality control takes place. Finally around 30 million observations are actively assimilated in 4D-Var, twice per day.

IFS currently produces approximately 86 TiB of model data per day, of which the largest portion (∼72

[35] Infrared Atmospheric Sounding Interferometer, IASI-Next Generation, Meteosat Third Generation - Infrared Sounder

TiB) is produced by ensemble forecasts. The next two resolution upgrades, even with conservative estimates in data growth, pose a significant data challenge in this regard. ECMWF also disseminates almost 30 TiB of products per day via ECPDS[36], a figure which has doubled since late 2017, and is becoming increasingly difficult to handle. The total data volume contained in MARS [37] exceeds 250 PiB.

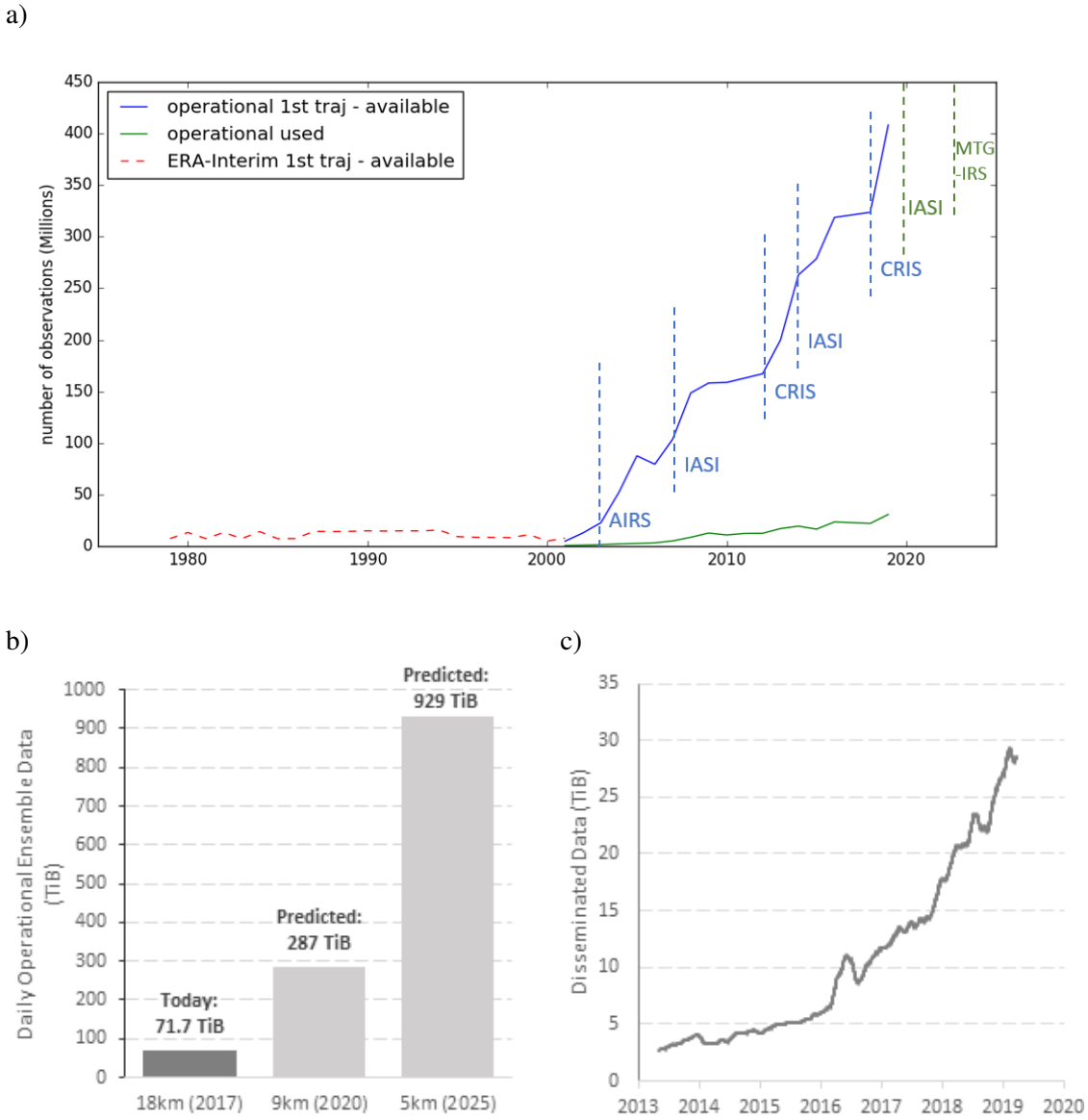


Figure 11: Time series of satellite observation numbers (a) monitored (solid blue) and assimilated (solid green) in the first trajectory of 4D-Var, ERA[38]-Interim monitoring (red dashed) is shown for comparison. Ensemble forecast model output (b, TiB/day) for model resolutions of 18, 9 and 5 km. Data volumes disseminated over the internet by ECMWF (c, TiB/day).

The work during the first phase of the Scalability Programme focused on the provision of key software

[36] ECMWF Production Data Store
[37] Meteorological Archival and Retrieval System
[38] ECMWF Reanalysis

components that alleviate primary data handling bottlenecks in the production workflow (Fig. 12). These developments helped to exclude costly processing tasks from the critical path (SAPP[39] and COPE[40]), to accelerate the model dependent quality control in the assimilation, and to manage observations and model fields in object-based data stores (IFS I/O server and FDB[41]).

In addition, a new data-centric simulator (*Kronos*) has been developed to replicate the combined operational and research HPC workload for testing different hardware infrastructures and supporting procurements. These developments were mostly supported by the ECMWF-internal COPE, Hermes[42] and OOPS projects (see Section 7.1).

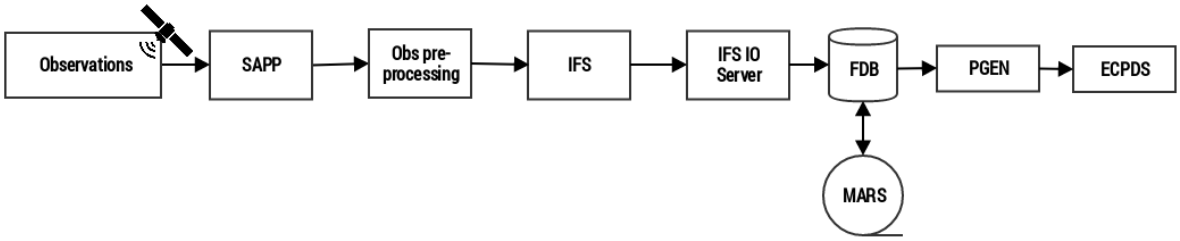


Figure 12: The current data pipeline in the forecasting system workflow: including pre-processing, screening and ingestion of observations; IFS data output handled by the I/O server; and harmonised data storage for model output, product generation and archival in tape archive via MARS.

Spin-off developments explored a distributed data-processing service, where requests for computations are farmed out to dedicated groups of compute nodes. A prototype of this distributed computing service was deployed on a Linux GPU cluster and tested with PGen[43], and it demonstrated that the PGen workload could be moved away from the HPC cluster. It also provided valuable technical guidance for the design of the new PGen software and the development of a flexible linear algebra back-end which was used in the new interpolation library MIR[44][72]. This allows MIR to use GPU hardware, and allows switching between BLAS [45] implementations optimised for different architectures (Intel MKL, Cray LibSci, etc.).

2.1.1 Pre-processing

The SAPP framework [44] has been operational at ECMWF since 2014. It has been further upgraded within the COPE project that was established to implement a new architecture where the observation pre-processing tasks are performed continuously as soon as the data arrive. The initial functionality of SAPP included the acquisition of observations from a large variety of sources (O(200)), the decoding from various formats (e.g. BUFR, GRIB, HDF, netCDF, text)[46], the application of initial quality control mechanisms and the conversion to a consolidated format for further processing.

[39] Scalable Acquisition And Pre-processing system
[40] Continuous Observation Processing Environment
[41] Fields Database
[42] https://www.ecmwf.int/en/newsletter/153/news/hermes-service-scalable-post-processing
[43] Product Generation; converts direct model output into data files according to user requirements
[44] Meteorological Interpolation and Regridding
[45] Basic Linear Algebra Subprograms
[46] Binary Universal Form for the Representation, GRidded Information in Binary, Hierarchical Data Format, Network Common Data Form

Integrated in the COPE pre-processing framework is an observation database (known as the ODB Store) that defines the interface between the continuous observation processing suite and the data assimilation and forecast suites. By decoupling the observation pre-processing from the rest of the forecast production chain, the overall resiliency of the system is increased. In this architecture, a failure in a single batch of observations no longer leads to a delay in forecast production. Issues can be investigated and remedied before the pressure of the time-critical path begins. In addition, the background-independent pre-processing of the majority of the incoming observations can be removed from the time-critical path. As the number and diversity of observations increases, the chances of corrupt or unexpected data exposing edge cases and causing failures in the processing systems increases. As our current processing chain is serial, failure anywhere in the chain is likely to lead to a delay in forecast production.

Substantial progress has been made in the implementation of the full COPE architecture. The pre-processing of conventional observations has been completely rewritten with the introduction of a new filter framework. Key components of the processing chain have been adapted to work with the new odb2[43] file format that allows observation storage in MARS. Over 90% of the observation pre-processing tasks have been adapted to run in a continuous processing framework. The progress culminated in a prototype suite running in real time for several months during 2018.

Importantly, the developments made for the COPE prototype were instrumental in the rapid implementation of the continuous data assimilation suite with model cycle 46r1 in June 2019 as this suite required continuous extractions from SAPP. The final stages of work allowing the full operational implementation of the COPE framework are to be completed after the activation of the HPC infrastructure in Bologna.

2.1.2 Observational data quality control

Data pre-processing needs to be able to deal with observational data source, distributor, format and metadata diversity, and to scale well across the expected compute and I/O node allocations.

The background-dependent quality control of observations is embedded in the data assimilation first model trajectory calculation. Over the years, the model resolution has increased but so has the number of observations to be compared to model grid points at observation locations. The efficiency and scalability of this process is constrained by the model itself and by the match-up process that requires efficient interpolation and load balancing as observation locations change all the time.

The scalability of both the first trajectory with data screening and the minimisation is shown in Fig. 13a. Following an end-to-end profiling exercise several aspects of the data screening were improved leading to much enhanced scalability (Fig. 13b).

In the current TCo1279 configuration, the direct cost of the observation I/O, observation operators and quality control in the first trajectory is now relatively small compared to the cost of the model integration. This is due to the fact that much of the observation processing is an embarrassingly parallel problem and has very good scalability characteristics. For example, the observation operators require no communication between MPI tasks. This is in contrast to the scalability of the model which is limited by the required communications of the spectral transforms. Over the last decade, the cost of the model has increased faster than the cost of observation processing (despite the large increases in observation numbers).

The remaining cost of the observation processing is dominated by those parts which require communication, namely the communication of model values to observation space and quality control aspects which require a global view of the data, e.g. background-dependent thinning.

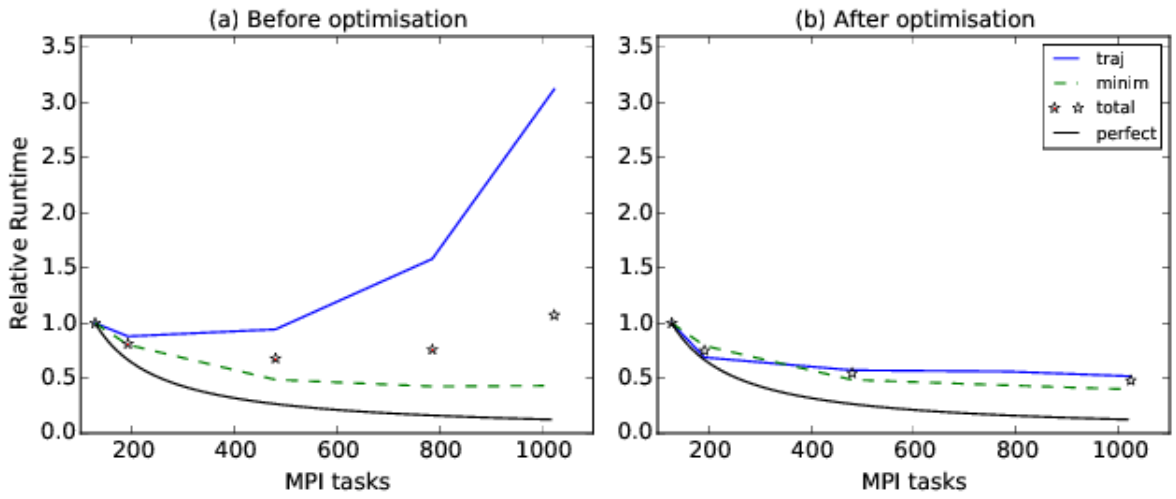


Figure 13: Strong scalability of model trajectory (incl. observational data screening) and minimisation before (a) and after (b) optimisation.

However, the TCo1279 configuration is not the only configuration run at ECMWF. For example, the EDA runs at much lower resolution and on far fewer nodes, but with the same number of observations. In addition, much of the HPC is filled with low resolution research experiments. In these configurations, the cost of the observation processing is still significant. A new initiative to refresh the observation processing codebase to refactor and in some cases rewrite key components of the system has been started. A new interface layer decouples the scientific code from the underlying in-memory and on-disk data storage. The overall aim is to allow ECMWF to benefit from the increasing diversity of observations expected in the coming decades. By making it easier for scientists to work with the code, new observation types can be tested and made ready for operational implementation sooner.

2.1.3 IFS I/O server

Most data output from the IFS is handled through an I/O server, which funnels data through dedicated I/O nodes to allow buffering and efficient marshalling of massively parallel data output. The I/O server was originally developed by Météo-France, and adapted to work with ECMWF's atmospheric (grid-point and spherical) fields for high-resolution forecast model output data. As part of the Scalability Programme, the I/O server has been extended to handle wave model and ensemble forecast data output.

As shown in Fig. 14, the wave model previously used an N-to-M output scheme, which aggregates fields on a designated subset of the model processes. This does not scale well due to global communications and the lack of asynchronicity when writing data. Adapting the wave model to use the IFS I/O server required significant effort in rearranging the initialisation of the wave model and adapting wave-specific GRIB encoding on the I/O nodes. It also required complex modifications for metadata transmission. Although the quantity of the wave model output is small compared to atmospheric output (approx 5%-20%), the number of fields written is very large, because two-dimensional wave spectra are written at every grid point. When the full spectra are written, data output is responsible for approximately half of the wall-time of the wave model, and the I/O server adaptations reduced this to almost zero. These wave model I/O improvements have been introduced into operations with cycle 46r1.

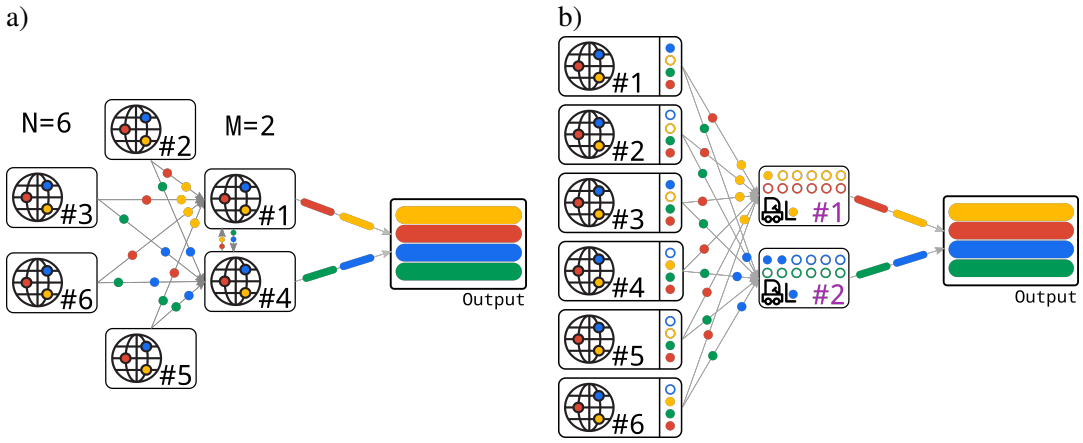


Figure 14: Previous (a) and new (b) wave model I/O handling. The previous N-to-M data output method suffered due to global synchronisation and the lack of buffering on the output nodes. The I/O server architecture allows the model to continue processing fields (colours) without blocking by I/O tasks (numbers).

The I/O server has also been validated for use with ensemble forecasts (both atmospheric and wave), allowing this technology to be leveraged for the most data-intensive output. This greatly helped in the operational deployment of cycle 45r1, in which the ensemble forecast runs were significantly affected by I/O.

2.1.4 Fields database

The Fields Database (FDB) is a software library, and an internally provided service, used as part of the ECMWF's weather forecasting software stack. It operates as a domain-specific object store, designed to store, index and serve meteorological fields as produced by the forecasting model. It acts as the first level of storage for recently created objects, efficiently receiving all of the model output and derived post-processing fields and making them available to the post-processing tasks in the forecast pipeline, as well as to users.

What is an object-based datastore?
Object storage, also known as object-based storage, is a strategy that manages and manipulates data storage as distinct units, called objects. These objects are kept in a single storehouse and are not ingrained in files inside other folders. Instead, object storage combines the pieces of data that make up a file, adds all its relevant metadata to that file, and attaches a custom identifier. This eliminates the tiered file structure used in file storage, and places everything into a flat address space, called a storage pool. This metadata is key to the success of object storage in that it provides deep analysis of the use and function of data in the storage pool[a].

[a] https://www.netapp.com/us/info/what-is-object-storage.aspx

The FDB serves as a hot-object cache in front of the more extensive MARS infrastructure. MARS[47] is a primary service offered by ECMWF that makes many decades of meteorological observations and forecasts available to a wide range of end users and operational systems. Around 80% of MARS requests are served from the FDB directly, typically for very recently produced data. A subset of this data is later re-aggregated and archived into the perpetual

[47] Meteorological Archival and Retrieval System

archive for long-term availability.

Every day, more than 200 TiB of data are written to and 370 TiB are read from the FDB, including both core operations and research activities. More than 100 TiB of this data is then moved to MARS for archival. At any given time, the total content of the operational FDB is estimated to be between 4 and 5 PiB.

The fourth version of the FDB (FDB4) was one of the most venerable pieces of software still in operations, with code containing references to the original Cray machine from more than 20 years ago. As a result, it had quirks that showed its age and restricted the design and operation of the forecast system. For example, FDB4 did not properly handle some error conditions, which could lead to corruption of indexing information or the data. This implied that when some forecast components had to be rerun, it required data to be wiped and the associated post-processing restarted. Further, only one forecast component could write to each database within the FDB4 at any one time without risking index corruption, which placed some constraints on the design of the forecast pipeline and suites.

To address these deficiencies, the development of the fifth version of the FDB (FDB5) began in 2015. This version brings several improvements over the previous version:

- Most importantly, FDB5 is now a transactional store, providing consistency and atomicity semantics in line with ACID[48] database principles. This means it is robust and resilient to failures, implying that a model or system failure does not corrupt existing data (as it did in FDB4), and that the model is able to restart where it left off. This saves critical minutes in the event of a catastrophic model crash and restart, minimising forecast delivery delays.
- By improving the consistency semantics, some restrictions on ECMWF's workflows have been lifted allowing for more flexible suite design. For example, FDB5 is no longer restricted to a single serial writer per database.
- Due to its tighter integration with the MARS service, the FDB5 now supports direct-to-MARS archival, which once adopted over all experiments and operational streams, will reduce overall congestion of the HPC storage system, by avoiding creation of temporary files in the archival process to MARS. Moreover, FDB5 has stronger verification of the data, and disallows fields which are not recognised by MARS.
- The FDB5 separates access via a configurable front-end API from storage back-ends. This builds a great deal of flexibility to develop and configure the system to make use of new storage technologies and paradigms without having to explicitly modify the forecasting workflow.
- The FDB5 should have slightly better performance for the same hardware, due to an improved indexing scheme for data retrievals. This improvement becomes more effective the larger the datasets become.

Another benefit of undertaking this massive transformation is that the I/O software stack is now more flexible and adaptable to new technologies. I/O systems have been evolving rapidly over the past 20 years and the pace of change is only increasing. ECMWF needs a way to support upcoming, possibly disruptive, technologies to allow us to build exascale forecasting systems in the future.

One example of novel storage technology was provided by the EU funded NextGenIO[49] project (see Section 7.1). Within this project ECMWF has developed a distributed front-end to enable the FDB to be

[48] Atomicity, Consistency, Isolation, Durability

[49] Next Generation I/O

provided as a service built on a cluster of nodes, and a back-end using Intel's Optane DCPMM[50], a form of NVRAM[51]. ECMWF demonstrated the first HPC run of a weather forecast using this new technology, showing that such a combination of software and hardware can support our scalability needs for the next decade, and provide performance enhancement by a factor of 10 compared to what which is currently achieved.

Bringing the newly developed FDB5 into operational use took some time. Existing components of the forecast pipeline depended on and were hard-wired to use the FDB4. In particular, the previous PGen software needed to be retired and replaced with a newer system before the FDB5 could be brought into time-critical operations.

2.1.5 Kronos Workload Simulator

The motivation for *Kronos* arose from the need for a more realistic representation of ECMWF's workloads in benchmark suites to be deployed on external clusters, for example, in procurements. In particular, the present benchmarks do not accurately represent data handling and disregard the extensive data traffic caused by research experiments.

Kronos is a suite of tools developed as part of the NextGenIO project for analysing and mimicking applications and workloads that run on HPC systems. These tools have been used to orchestrate the large-scale profiling of forecast suites run at ECMWF, and to support the analysis of the resulting data for workflow optimisation. For the above reasons, there is a focus on the role of I/O in multiple-job workloads.

The workflow simulator uses the concept of an abstract workload that is expressed only in terms of measurable invariants, namely the amount of work that is performed rather than the number of processes, distribution of this work across processes and nodes, or the time and resources taken to complete it. Once this description has been obtained, it is possible to execute the workload in a range of contexts to determine how well the hardware and software environment supports the workload.

The *Kronos* executor is a small, event-driven workload manager. Given an input description of work it submits jobs to the supercomputer via the scheduler, and is notified of their progress and completion (see Fig. 15). These jobs can be composed of a *Kronos* synthetic application, which is a small executable that can be configured to run a series of configurable kernels that mimic the computing and data handling behaviour of a real application. Alternatively, these jobs can be real applications themselves.

Both time-critical operational and research workflows are large, complicated and composed of a large number of sub-components. For their representation by *Kronos*, these workflows have been broken up into a very large number of separate tasks that are submitted to the scheduler (and monitored) by ECMWF's suite manager *ecflow*. The structure of these suites largely shadows the true data dependencies that exist between the constituent tasks, but these dependencies are ultimately implicit in the more explicit task-graph.

For monitoring the corresponding data movements and dependencies, pre-operational suites (e-suites) and research suites were instrumented with a range of profiling tools (specifically Darshan, IPM [52] and ARM MAP). As *Kronos* contains tools for parsing, standardising and combining the outputs from these tools it is able to translate this to input for the workflow simulator, and to generate modular and scalable

[50] Data Center Persistent Memory Modules
[51] Non-Volatile Random Access Memory
[52] Integrated Performance Monitoring

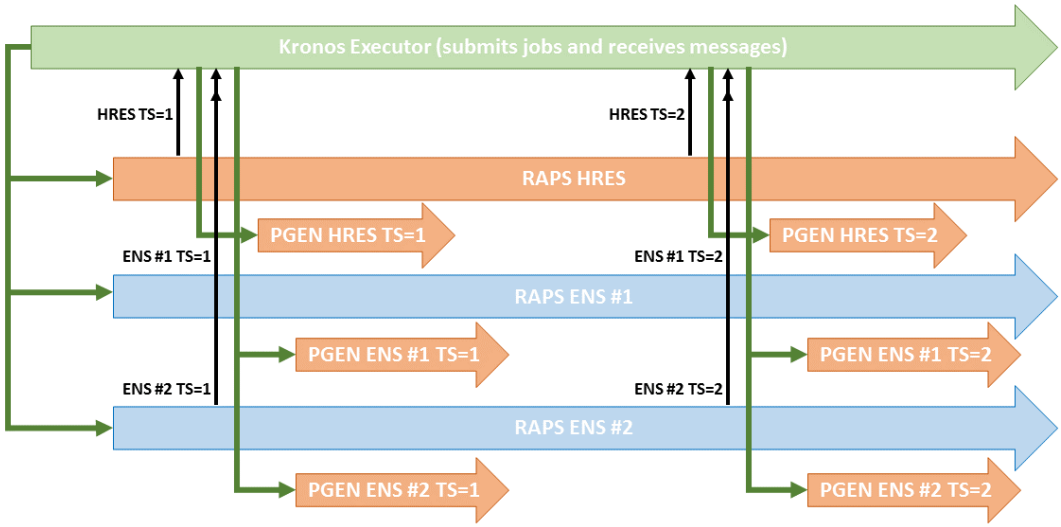


Figure 15: The *Kronos* executor executes parts of a workload mimicking components of time-critical operational suites. As in operations, it includes a number of model tasks (both HRES and ENS) run in parallel, and launches appropriate PGen tasks when intermediate model execution steps have completed.

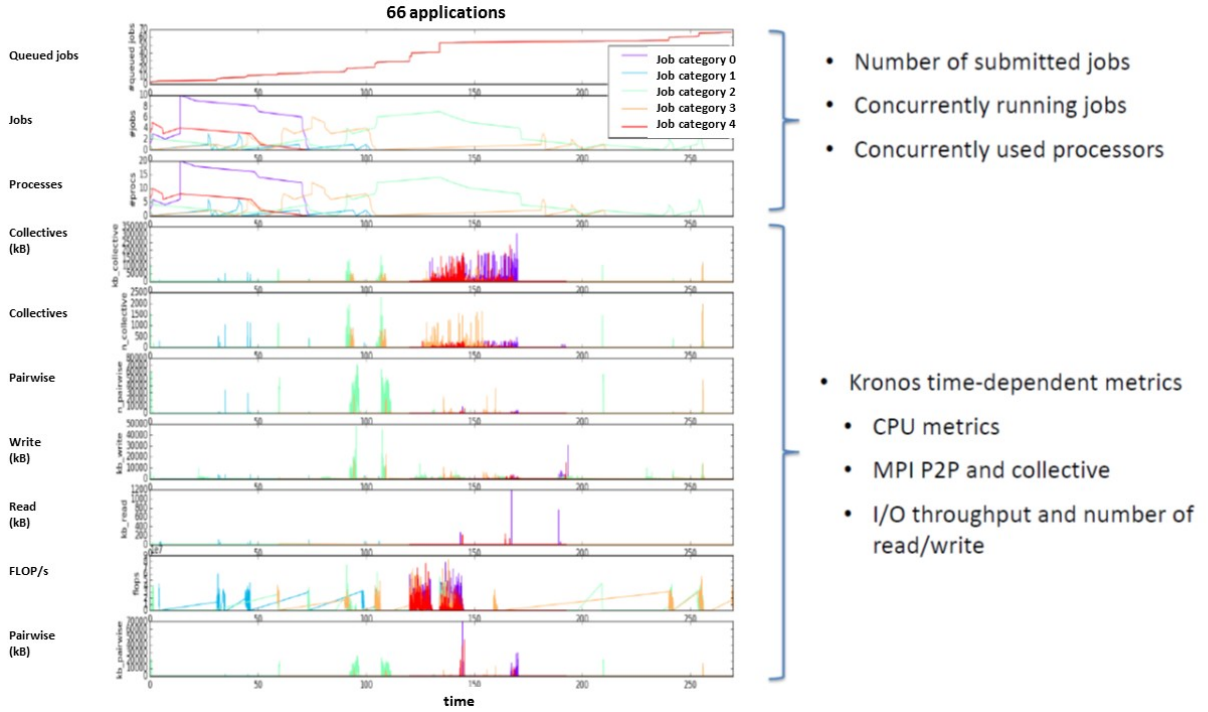


Figure 16: Schematic example of *Kronos* simulated workload profile for computing and data handling. Top three panels refer to job categories (queued and running jobs, concurrently used processors), bottom panels display performance metrics: data communication (volume (kB) and number of collectives), pairwise interaction and volume (kB), write and read volumes (kB), and FLOP/s. Graphs are for illustration purpose only so that complexity of workload for (here) 66 applications can be visualized.

workflows mimicking the performance of the real workflows. Fig. 16 shows an example of simulated

data movements.

Given the importance of data handling for benchmarks, Kronos was also employed in the HPC2020 procurement, and deployed on several vendor machines. The results formed the basis of the committed performance projections for the new machine to be installed in Bologna.

2.2 Data assimilation and forecast model

Historically, the IFS has been the central software repository for all components of the model, the data assimilation system and the supporting libraries. Other models like WAM[53], NEMO[54], LIM[55] and HTESSEL[56] have been associated with the IFS even though they are separate developments and, in the case of NEMO and LIM, community models developed by external groups. The COMPO suite is another example of an external model code that has become an integral part of the IFS. Often these different components are also required to run stand-alone rather than in coupled mode.

As the main scientific strategies of both data assimilation and forecast model have been presented in past papers [109] [13], the focus here will be on those developments that relate to the computational performance. This strongly interfaces with scientific considerations though, a theme that will play through the entire document as there are many points where scientific and computing performance are traded off against one another or where advances in compute performance require algorithmic changes.

2.2.1 Data assimilation

Data assimilation is an essential component of a numerical weather prediction system. ECMWF has developed a world leading state of the art hybrid 4D-Var (hereafter called 4D-Var) data assimilation system. 2017 marked the celebration of the 20th anniversary of 4D-Var in operations.

4D-Var: The continued success of the method stems from its favourable properties that allow to use a very large volume and diversity of observations also from cloud-affected areas, a very comprehensive set of tangent-linear versions of model dynamics and physics parametrizations, and flow-dependent background error covariances produced by the EDA.

4D-Var corrects a short-range forecast trajectory with all available observations in the assimilation window, in order to produce the best estimate of the current state of the atmosphere. The algorithm makes use of observations in a way that is consistent with the model dynamics and physics while accounting for the quality of the observations and the model itself. In practice, the analysis is obtained by minimising a non-linear cost function using the Gauss-Newton method, which results in a sequence of computationally tractable quadratic minimisations. Quadratic cost functions are minimised using iterative Krylov solvers.

The key to 4D-Var's success is in using the tangent linear and adjoint models to propagate in time the background error estimates (the errors of the prior information) which are only provided at the start of the assimilation window. This is different from ensemble methods, like EnKF[57] or 4DEnVar[58], which con-

[53] Wave model
[54] Nucleus for European Modelling of the Ocean
[55] Louvain-la-neuve sea Ice Model
[56] Hydrology Tiled ECMWF Scheme of Surface Exchanges over Land
[57] Ensemble Kalman Filter
[58] 4D Ensemble Var

struct the 4-dimensional prior forecast covariance using the ensemble of non-linear forecast trajectories. The latter are more prone to sampling errors and require spatial localisation in the case of EnKF or both spatial and temporal localisation in the case of 4DEnVar/4D-EnKF, unless they employ large ensembles, for example, at lower resolution to generate sufficient sampling.

Other strong points of 4D-Var also include its ability to account for weak to moderate non-linearities thanks to its incremental formulation with multiple outer loops, its ability to perform global analyses (avoid restricting influence of observations through localisation), variational bias correction and variational control of gravity waves. However, some of these features are not available by construction but because significantly more effort has been spent on the optimisation of 4D-Var[70, 71].

On the other hand, 4D-Var in its current configuration, is an intrinsically sequential algorithm, both at the outer loop and inner loop level. The outer loop iterations produce increasingly accurate linearisation trajectories starting from the previous outer loop initial state and the initial analysis increments computed in the inner-loop minimisation. Similarly, each iteration of the minimisation algorithm requires the sequential execution of the tangent linear and adjoint models and all the iterations are sequential.

The parallelisation of 4D-Var has up to now been approached by parallelising the models used in the algorithm (non-linear, tangent linear and adjoint) in the spatial domain. In this context, it should be noted that the strong and weak scaling properties of the tangent linear and adjoint models are comparable to that of the non-linear model for the same spatial resolutions. In practice it would be too costly to perform the minimisations at the same resolution as the non-linear model, even though it would result in a better quality of the analysis. Thus, a multi-incremental algorithm is employed in practice, where the sequence of quadratic problems is solved at lower resolutions than that of the nonlinear model used in the computation of the outer-loop trajectories. Lower spatial resolution, however, limits the amount of parallelism achievable in the minimisations compared to the high-resolution, non-linear model trajectory simulations.

It is becoming apparent that increasing the resolution of the minimisations in the future cannot be exclusively addressed by the spatial parallelisation. Higher resolution of the minimisations has twofold consequences on the total run-time. The time step of the tangent linear and adjoint models needs to be reduced and the convergence of the Krylov solver becomes slower. Assuming that the number of grid points per core was kept constant for higher minimisation resolutions, the minimisation run-time would not fit into tight operational schedules. Increasing the spatial parallelism would not be sufficient due to increasing communication overheads. With hardware architectures becoming ever more parallel and with individual cores not getting faster, there is a clear need to develop more parallel data assimilation algorithms which better map to the available resources. While the constraints of a time critical path have been to some extent relaxed through the introduction of the recently implemented continuous data assimilation approach [69], the underlying scalability bottleneck of 4D-Var will eventually need to be addressed.

OOPS: Data assimilation had historically been an integral part of the IFS. While such approach seemed natural at the time when 4D-Var was being developed, the system quickly outgrew its original design and became difficult to maintain and modify. As a consequence, it was difficult to prototype and quickly test new ideas like the generalised weak-constraint and the saddle-point algorithm in the recent past. Also the complexity and obscurity of the code base constituted a high initial barrier for new developers and made it difficult to pursue external collaborations. These considerations led to the idea of developing a new data assimilation framework based on an object oriented software paradigm.

With the development of OOPS[100], a more generic top level has been created that mainly serves the

generation of the initial conditions but that can also be considered as a driver that can configure forecasts with and without coupling even though OOPS will not immediately be run in operations as the top-level driver infrastructure for forecasts. It is clear though that OOPS does execute model trajectories for the comparison between short-range forecasts and observations, and that this will be performed with a fully coupled system in the future.

OOPS has caused a substantial rewrite of the IFS in order for the IFS code to map reasonably to the OOPS object classes and methods. OOPS represents the model in a non-global view, thus it is cloneable at a different resolution or with a different functionality in the wider workflow. In the long term, OOPS therefore will constitute the topmost control layer for a range applications including the forecast, data assimilation using a variety of algorithms, SVD[59] [51], FSOI[60] [10] and unit tests.

OOPS was started before the inception of the Scalability Programme, but its main objective - adding algorithmic flexibility thanks to a modern, object-oriented software framework to combine good science and computing performance - lies at the heart of the programme. Hence OOPS became part of the programme and was endowed with its resources. On the model side, algorithmic flexibility is key as well for better computational efficiency and enhanced physical realism towards higher spatial resolution and more model complexity.

The main benefits of OOPS were anticipated to be:

- modernised, easier-to-read code with fewer inter-dependencies from one area of code to another, potentially making code more robust and easier to debug;
- the ability to test new variational assimilation algorithms easily e.g. 4D-EnVar, parallel minimisation (saddle-point was the favoured option at the time [41]), with a view to improved science, scalability and capability;
- a unified framework for data assimilation across multiple Earth-system components with a view of developing a coupled data assimilation system;
- reduced set-up and I/O costs leading to efficiency and scalability gains;
- a unit testing framework to improve system resilience;
- fast prototyping of new ideas;
- to develop other applications: SVD, FSOI etc.

The key idea behind OOPS is to decouple algorithmic aspects of variational algorithms from model specific implementation of the operators used in those algorithms.

The design of OOPS comprises three layers (Fig. 17). In the middle layer, OOPS defines a set of templated abstract interface classes, referred to as building blocks. They represent entities manipulated by variational algorithms like state, increment, control vectors, covariances, model, observation operators. The building blocks are combined together by the top layer into applications, like 4D-Var, forecast, FSOI, etc. Abstract interface classes wrap around model-specific implementations of those classes through the C++ templating mechanism. Each abstract interface class receives a model as a template argument. A model template argument defines a set of aliases allowing to link generic classes to model-specific implementations which amount to the bottom layer of the system. The link only happens at compile time. This allows the separation of the control top and middle layer from the bottom model-specific layer.

[59] Singular Vector Decomposition
[60] Forecast Sensitivity to Observation Impact

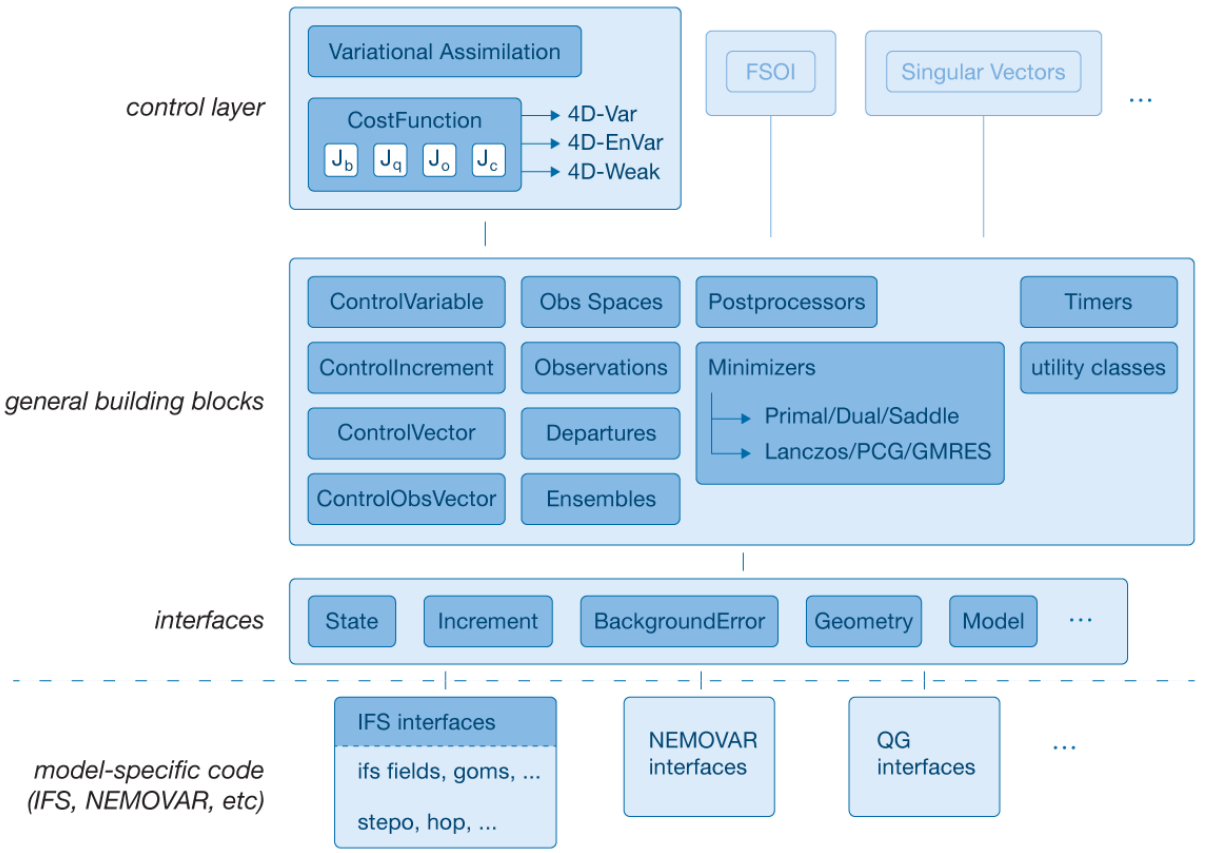


Figure 17: Conceptual design of OOPS with the application top layer, the templated building blocks of variational assimilation in the middle layer and the interfaces to model specific components in the lowest layer. The main user-exposed features are implemented via inheritance represented in blue. Ensemble algorithms could be implemented as yet another application in the future. They would require, while sharing a subset of interfaces and general building blocks, a completely new design of the control layer. The faint-coloured elements (FSOI and SVD) represent additional functionality, and are not essential for producing the analysis.

The OOPS templated design comes with advantages and disadvantages. OOPS is templated for a single model, the IFS. This will need to be revised for implementation of coupled data assimilation, for example. An alternative abstraction mechanism could employ polymorphism; however, it would have a significant disadvantage of requiring dynamic casting which comes with no type-safety guarantees. The key advantage of the OOPS design lies in the anticipated invariance of the middle layer: the building blocks. It is highly unlikely that the interfaces will change significantly in the future. The topmost layer is relatively lightweight and can be quickly re-designed to meet future requirements. It should be noted that although OOPS was originally conceived for variational algorithms it could be extended to encompass a family of ensemble algorithms if needed. OOPS could then become a common platform for both variational and ensemble algorithms facilitating development and maintenance of hybrid systems.

Status: The restructuring of the IFS code required to make OOPS work with IFS involved changing a large majority of the IFS variables into objects, implemented as Fortran derived types, and grouping these derived types hierarchically to match the OOPS interface classes. The IFS subroutines were then

modified to receive the restructured variables by argument, thus removing the hard-coded global view of variables that existed previously. Further work is required for the atmospheric composition suite COMPO in OOPS mode, because the OOPS development has not considered composition so far. There may be interesting science perspectives for, coupling, trace gas and aerosol assimilation, such as the introduction of surface fluxes in the control variable or the optimisation of reaction rate parameters, which could be approached with OOPS in an easier way than before.

Similar effort has been undertaken to adapt the ocean component to OOPS in view of developing the future coupled data assimilation system. The NEMOVAR[61] code has already been separated from the NEMO code base, refactored and interfaced to OOPS. The NEMO model itself has not been refactored and ported to OOPS yet, due to the external dependency on the NEMO community. The NEMO model will eventually need to be adapted to OOPS, if the future coupled data assimilation system is to be flexible. This is also required to retain the efficiency gains resulting from running non-linear trajectories and minimisations from a single executable.

The considerable amount of work that went into refactoring IFS and NEMOVAR has allowed to modernise the ECMWF data assimilation codebase by making it more modular and more flexible. The work is not yet complete though. Outstanding points include the development of a unit-testing framework allowing to exercise different components of the system independently and therefore facilitating maintenance, code development and integration, as well as long-term system sanity. Not every component of the IFS has been implemented in OOPS either.

New applications for the computation of singular vectors and FSOI will need to be developed. The task should be facilitated by the ability to leverage to a large extent the already existing infrastructure. The more challenging task will be to adapt the system for coupled data assimilation. Some of the design choices may need to be re-evaluated in the future, like the idea of running the entire multi-incremental 4D-Var from a single executable. In the context of strongly coupled data assimilation, questions also arise regarding the future of land-surface data assimilation, which currently employs a SEKF[62] filter. A thorough discussion of future plans is deferred to Section 4.2.

The integration of IFS into OOPS (OOPS-IFS hereafter) is now largely completed. The only remaining component that has not yet been interfaced is the weak constraint term. The task is recognised as a technical task of moderate complexity and will be carried out after the finalisation of IFS cycle 47r1, in Q3 2019. OOPS-IFS is capable of reproducing the current operational HRES configuration, however a few subtle differences still exist. The main difference is in the treatment of the all-sky observations. IFS currently does an additional screening of all-sky observations in the non-linear observation operator in each outer loop. This feature will likely be deprecated in future and therefore was not implemented in OOPS-IFS. The EDA configuration will require minor technical work in order to be able to warm-start the limited memory preconditioner using the Ritz vectors from the control EDA member. Additional work will be required for the implementation of the continuous data assimilation, where the screening is done during each outer loop. While such a scenario is already supported by OOPS, the fact that all the outer loops are run from a single executable will require devising a way to synchronise OOPS-IFS execution with the availability of new observations.

Performance: Fig. 18 shows the strong scaling of OOPS-IFS and IFS for the TCo1279 configuration. The anticipated efficiency gains resulting from running the entire incremental 4D-Var from a single executable amount to roughly 20% run-time reduction for the operational configuration (352 nodes)

[61] NEMO variational data assimilation system
[62] Simplified Extended Kalman Filter

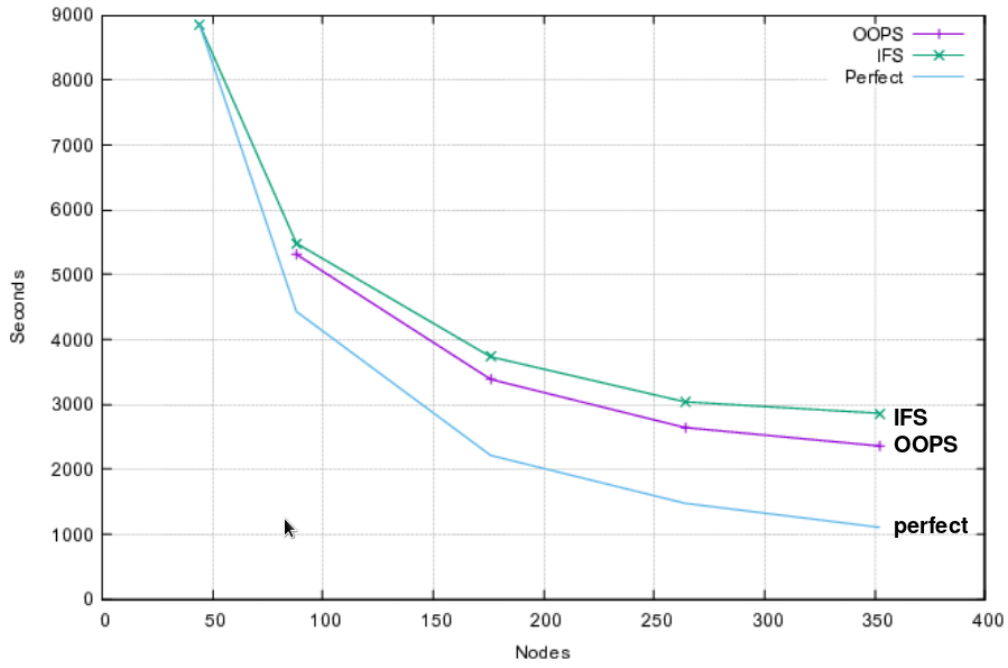


Figure 18: The cost of setting up operators is constant irrespective of the number of processors. Comparison of strong scaling of IFS and OOPS-IFS 4D-Var.

compared to the IFS. However, the OOPS-IFS restart mechanism, allowing to start the algorithm from the last completed outer loop in case of a crash, has not been active in the tests, which may to some extent limit the efficiency gains. It should also be noted that the OOPS-IFS system has not been properly profiled and optimised yet. Ongoing work suggests that the efficiency gains may be noticeably higher.

OOPS has already proved to be a very useful tool to quickly prototype new ideas and test them with simplified models like the Lorenz-95 model and the QG[63] model. The generalised weak-constraint formulation has been implemented in OOPS. This algorithm was designed to open the 4D-Var time dimension to parallelisation with the saddle-point formulation. Recent results of [47] and [48] have demonstrated that the algorithm converges non-monotonically and is poorly conditioned. Research on novel solvers ensuring monotonic reduction of the quadratic cost function is being carrying out by external partners (CERFACS) and their preliminary results are encouraging. In parallel, new algorithmic approaches based on the randomised SVD concept are being explored.

Work has also started on adopting the CAMS-system to OOPS. New ideas, like running multi-resolution coupled atmosphere-chemistry model leading to significant efficiency gains will be explored, and require adapting *Atlas* [26] (see Section 4.2). There is also an ongoing effort on the implementation of flux inversion techniques for anthropogenic CO_2 emissions for the CHE[64] project funded by the European Commission.

[63] Quasi-Geostrophic
[64] Carbon Human Emission

2.2.2 *Forecast model*

Enhancing model complexity where it improves forecast skill includes fully coupled simulations of the atmosphere with ocean, sea-ice, wave and land surfaces including lakes and snow, and the addition of chemical processes.

Key physical processes that drive global circulation, like deep convection in the tropics and meso-scale eddies in the ocean, are limited by spatial and temporal resolution. Moreover, there is a need for a sufficiently large ensemble size so that complex probability distributions of extreme forecast features can be sampled well enough. However, ensemble size and better spatial resolution as well as enhanced model complexity translate immediately into significant computational challenges [96], of which resolution is the most demanding because a doubling of resolution roughly translates into eight times more computations due to corresponding reductions in the time step.

The scope of the activities on numerical methods and technical model development is to fundamentally establish flexibility, with new programmatic interfaces and data structures, together with exploring scientific choices that open up new possibilities for a complete description of the Earth system. These novel choices need to scale better and use future hardware in an energy-efficient manner [109]. Substantial investment has been made in the past four years and this section describes the progress of this work.

IFS core model: ECMWF's objective is to develop one of the most advanced and flexible model infrastructures for operational, global NWP [65] applications being able to optimally exploit future computer hardware to be released over the next 20 years. In addition, the code infrastructure should facilitate efficient collaboration with ECMWF Member States, both for scientific exploitation and for the effective use of boundary conditions from ECMWF for limited-area modelling.

The spectral-transform IFS (IFS-ST) model is based on the hydrostatic equations, with a non-hydrostatic option. Horizontal discretisation is realised on the octahedral reduced Gaussian grid. Advection is solved using the semi-Lagrangian method, combined with a semi-implicit solution procedure that requires a solution of a Helmholtz equation arising from the linear implicit terms, which is solved in global spectral space. The latter requires the direct and inverse spectral transforms at every time-step with data-rich MPI communications, but adds the convenience that all the required horizontal derivatives and the Laplacian can be calculated with high accuracy in spectral space.

The vertical discretisation uses finite elements, which provides more accurate solutions for the required vertical integral operators in the hydrostatic model. The overall solution procedure is extremely stable with large time steps, and is instrumental to achieving a short time to solution. Coupling to the physics is arranged along the space-time integral of the semi-Lagrangian trajectories. Tangent linear and adjoint models exist that are closely aligned with the non-linear IFS. Different domain decompositions in grid-point and spectral space assure fast parallel computations in either space, with bottlenecks in the halo updates for the semi-Lagrangian and the transpositions from grid-point to spectral space and back [109].

A number of key areas have been addressed where the current IFS does not allow sufficient flexibility and hence hinders progress in adapting to future computing architectures. In order to substantially increase the readiness level for new technologies, we seek alternative numerical algorithms that depend on fundamentally different communication patterns, both local and global. It is likely that different numerical algorithms will be justifiable and efficient in terms of time-to-solution but that they will not be competitive in terms of energy to solution. The latter is a growing problem with rising energy costs absorbing

[65] Numerical Weather Prediction

substantial parts of the compute budget in weather prediction services.

In global NWP, finite-volume methods appear to emerge as one of the standards of atmospheric modelling towards non-hydrostatic scales, with a number of other weather forecasting centres employing a finite-volume approach on alternative grids. Examples are FV3 (NCEP), and in some parts LFRiC (UKMO), with both using a cubed-sphere grid, and ICON (DWD, shared with MPI) using an icosahedral grid. Finite-volume models are further used at research institutions, such as MPAS (NCAR), DYNAMICO (IPSL) and NICAM (RIKEN); see [104] for an overview of recent atmospheric dynamical core formulations.

By developing a hybrid and hierarchical methodology that continues to support the spectral transform technique, ECMWF has introduced added flexibility through the development of the IFS-FVM [97, 65]. The IFS-FVM dynamical core incorporates finite-volume, semi-implicit integration methods for atmospheric dynamics that is suitable from planetary- to micro-scales. Essential aspects of the IFS-FVM discretisation are local computational stencils and low-volume parallel communication that occurs predominantly with the nearest neighbours, which is fundamentally different to the parallel communication in IFS-ST.

The finite-volume semi-implicit integration of IFS-FVM combines flux-form Eulerian advection based on the non-oscillatory MPDATA scheme (see [66] and references therein) with three-dimensional, fully implicit time-stepping for all acoustic, buoyant and rotational modes. Overall, this formulation represents a good compromise between robustness and accuracy for long time steps on the one hand and local computational patterns on the other, while at the same time providing fully conservative and monotone advective transport. The default three-dimensional, implicit time stepping is not mandatory as in IFS-ST, and optional switches for explicit treatment of selected processes will be further advanced if beneficial for future HPC infrastructures.

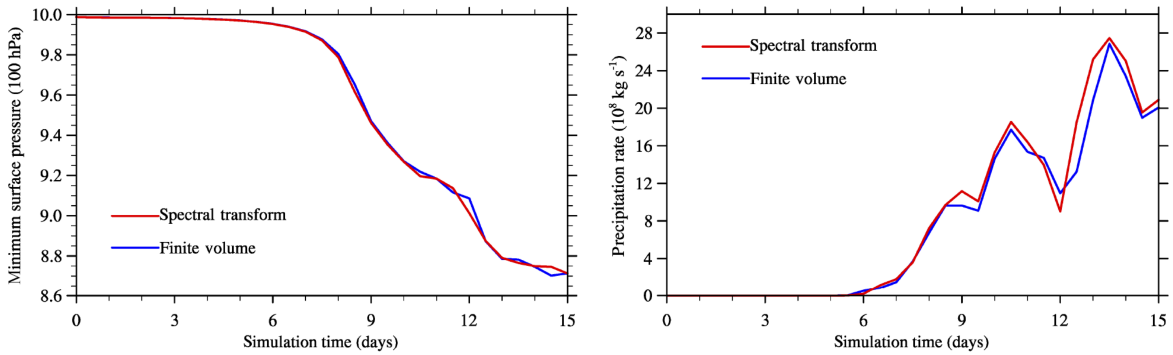


Figure 19: Time series of minimum surface pressure (left) and area-integrated precipitation rate (right) for a 15-day baroclinic instability simulation with IFS-ST (red) and IFS-FVM (blue), both run on the same O320/TCo319 horizontal grid and with the same IFS cloud parametrization; from [65].

Other advantages of IFS-FVM are flexibility with regard to the choices of non-hydrostatic governing equations (either fully compressible or soundproof, optionally combined with either deep- or shallow-atmosphere and different perturbation formulations) as well as horizontal and vertical meshes (quasi-uniform or adaptive/variable resolution). The second-order, median-dual, finite-volume approach on the co-located grid and the height-based terrain-following vertical coordinate in IFS-FVM, together with the employed data structures, programming paradigms and abstractions provided by the *Atlas* library are expected to greatly facilitate adaptation and performance portability with regard to novel, low-energy

heterogeneous HPC architectures.

The IFS-FVM is attractive from a physical point of view as it ensures local and inherent conservation of the mass of air and its constituents. This is an important property for atmospheric composition forecasts but also for cloud microphysical processes at resolutions sufficiently high to explicitly resolve deep convection. Furthermore, IFS-FVM is robust with respect to very steep orographic slopes. The local computational patterns of IFS-FVM, centred around the co-located grid, are reapplied uniformly in all computationally intensive parts of the code. These properties facilitate cache-efficient programming and are advantageous for implementation with a DSL. The implementation may be further optimised for meshes that permit direct (versus indirect) access of neighbours in the spatial discretisation. Last but not least, the IFS-FVM can operate on the same octahedral reduced Gaussian grid and with the same IFS physical parametrizations, which is beneficial for general code maintenance [65] as well as for future data assimilation efforts (that may still require spectral transforms in the computation of the background error statistics for example).

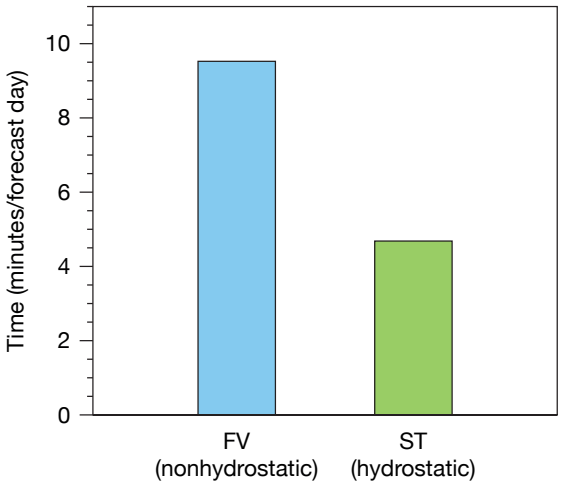


Figure 20: Elapsed time to run one day of the dry baroclinic instability benchmark with the non-hydrostatic IFS-FVM and hydrostatic IFS-ST, using the same O1280/TCo1279 horizontal grid with 137 vertical levels and 350 nodes (12,600 cores) of ECMWF's Cray XC-40 ; from [67].

Previous work has verified the solution quality of IFS-FVM for relevant test cases of intermediate complexity against the established IFS-ST as well as other models including those with higher-order local discretisation methods in the context of DCMIP; see e.g. Fig. 19 and [65, 113]. A snapshot of the computational efficiency of the non-hydrostatic IFS-FVM with the flux-form Eulerian advection, in comparison to an equivalent configuration of the hydrostatic IFS-ST with the semi-Lagrangian advection, is shown in Fig. 20 and appears promising. While both models in Fig. 20 still used double precision in the computations, the development of a single-precision IFS-FVM was completed in December 2019 and will become the basis for future comparisons.

In 2019, the first real weather re-forecasts were performed with IFS-FVM coupled to the full IFS physical parametrization package. Further refinement and more comprehensive experimentation will be conducted in 2020.

Atlas: Fostered by externally funded activities in ESCAPE[66], the flexible and dynamic data structure framework *Atlas* [26] has been developed. *Atlas* provides mesh generation and function spaces, which in return provide flexibility for the discretisation of fields on which alternative algorithms can operate.

Based on the *Atlas* functionality, the alternative IFS-FVM dynamical core uses entirely different computational and communication patterns, and substantially reduces global communication. The *Atlas* library also forms the basis for many tasks in the product generation and the post-processing engine of MARS (through MIR) and provides future opportunities for all relevant model regridding operations to observa-

[66] Energy-efficient Scalable Algorithms for Weather Prediction at Exascale

tion locations in data assimilation.

Atlas developments are a key activity in ECMWF's Scalability Programme in order to abstract concerns of data management, memory layouts and parallelisation from scientific model developments. The goal is to have *Atlas* serve as a common foundation between Earth-system model components and pre- and post-processing software, thus creating a focal point where data structures are managed and easily adapted to hardware such as GPUs. Because *Atlas* is primarily written in C++, it has ready access to programming models and libraries otherwise not available to Fortran. *Atlas* can e.g. directly use the CUDA [67] programming model for GPU adaptation, or libraries like Kokkos [37] that abstract parallel loop execution for various hardware architectures. The C++ implementation makes it also useful to serve as a library for use in a domain-specific language as is being developed in ESCAPE2. *Atlas* provides Fortran interfaces to enable integration into the existing IFS.

While continuing with the hybrid parallelisation model of OpenMP and MPI, efforts are underway to use *Atlas* for managing selected data structures in the atmosphere model of the IFS. This is necessary to facilitate the introduction of the optional IFS-FVM into the cycle-based IFS code framework. Under development is the integration of ESCAPE advection schemes into IFS, which are based on *Atlas*. These advection schemes are capable of performing advection of tracer fields on a different (coarser) grid than the atmospheric model grid, and using alternative numerical techniques, e.g. based on conservative finite volume discretisation. *Atlas* helps to minimise parallel communication by ensuring the advection scheme's grid follows the same domain decomposition as the atmospheric model grid. Within the IFS an abstract advection interface has been developed that delegates execution to the scheme. This is depicted in Fig. 21.

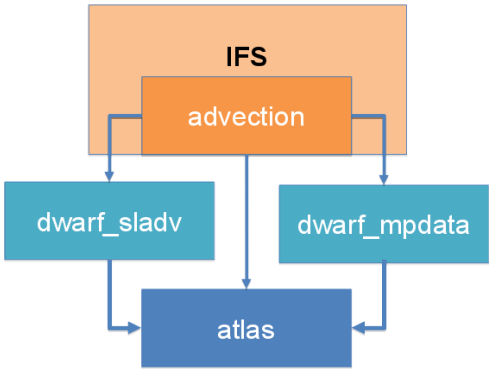


Figure 21: *Atlas* serves as a common foundation between two ESCAPE advection schemes (*dwarf_sladv* = semi-Lagrangian for IFS-ST and *dwarf_mpdata* = MPDATA for IFS-FVM). An abstract advection interface within IFS, also based on *Atlas*, delegates the execution to either of these schemes.

When *Atlas* becomes responsible for IFS's data structures, parallelisation, and data layouts in memory, it becomes the ideal framework to cater for the remapping of fields between Earth-system model components. This is schematically shown in Fig. 22. The proof of concept has been implemented with the two above-mentioned ESCAPE advection schemes. *Atlas* abstracts the remapping implementation. Current interpolation methods available encompass linear-, cubic-, quasicubic-, and *k*-nearest neighbour interpolation. Other interpolation methods such as conservative interpolation are in development. These interpolation methods, incorporated in the *Atlas* library, are thus available to a multitude of applications, including ECMWF's MARS and the PGen suite.

Status: As illustrated before, the continuous effort to improve the performance of the spectral transforms, combined with the trend towards fewer but more powerful nodes and the use of several accelerators per node, has kept the spectral transforms competitive for global hydrostatic simulations. This has been achieved by combining the recent advances on spectral-transform computational speed-ups with the use of the cubic-octahedral, reduced Gaussian grid [108] [109] [73], the use of the Fast Legendre Transform (FLT) [108], the use of single precision [106] and the adaptation to GPUs [82]. In this con-

[67] Compute Unified Device Architecture

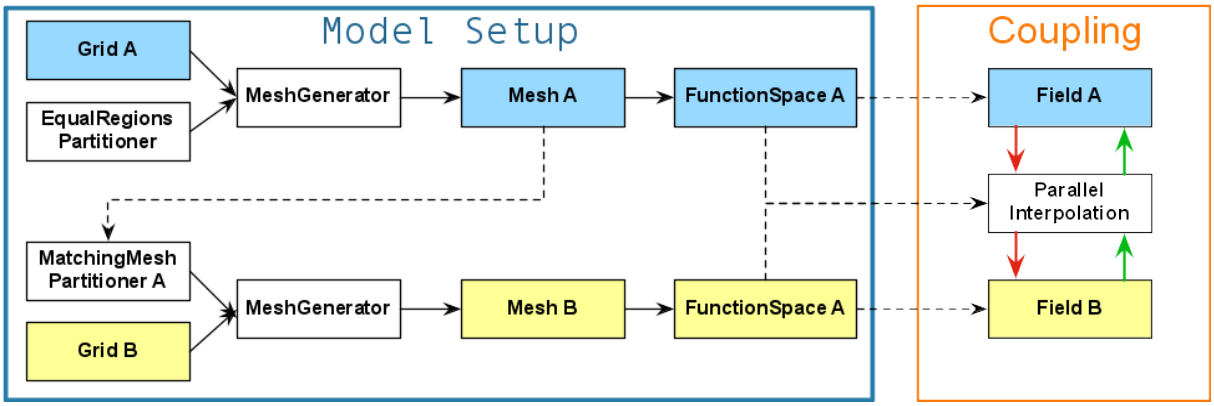


Figure 22: Two Earth-system components (indicated by blue and yellow) could be set-up using *Atlas*, where the grid of one component follows the domain decomposition of the grid from another. *Atlas* then provides parallel interpolation operators to remap fields between the components as required.

text, the implication of using a specialised spherical harmonics co-processor has also been explored. However, given the strong dependence on the communication bandwidth and not the compute performance for global spectral transforms in time-critical applications, a spherical harmonics co-processor is not considered useful and possibly creating a bottleneck with increased resolution.

The continued competitiveness of spectral transform methods has led to the conclusion that this method is unlikely to be abandoned before also the hydrostatic assumption in the governing equations becomes problematic in global simulations. However, communication and energy cost is a driving concern and justifies the exploration of alternative algorithms with different (i.e. more local) communication patterns, in particular when moving towards very high-resolution, non-hydrostatic global simulations of O(1km) or better, and considering exascale computations on a variety of the new HPC architectures. As a result, there is a continued research effort and need in pursuing algorithmic approaches that simply do not communicate beyond a certain halo size while retaining other favourable properties such as the large time step in the semi-implicit semi-Lagrangian integration scheme.

In this direction, as a new trend in NWP modelling, the first models based on a higher-order local Galerkin dynamical core are appearing and starting to reach operational readiness. Examples are the US Navy Research Lab's NEPTUNE[68] that is based on the NUMA[69] model [29], [76] and the KIM[70] system for global weather forecasting [21, 57].

One of the main reasons behind the increasing interest for such methods is that spectral element and discontinuous Galerkin methods share a key ingredient of success of the models developed by ECMWF, namely higher-order numerics, while unlike the spectral transform technique, they only require a small communication stencil, thereby making them more local in nature and hence more amenable for use on novel hardware such as accelerators.

However, although these new methods offer the means to represent spatial differential operators with high fidelity and robustness while also exhibiting characteristics that make them amenable to high performance on exascale systems, they tend to be relatively expensive and suffer from severe stability restrictions when coupled with explicit Eulerian time integrators.

[68] Navy Environmental Prediction System Using the NUMA Core
[69] Nonhydrostatic Unified Model of the Atmosphere
[70] Korean Integrated Model

Within the externally funded ESCAPE-2 project, ECMWF is now exploring a numerical formulation which aims at retaining the benefits from both IFS-ST and IFS-FVM, namely higher-order accuracy combined with long time steps for efficiency and data locality with geometric flexibility, respectively. In this spirit, ECMWF is exploring a modal (Legendre polynomial based) discontinuous Galerkin (DG) formulation, coupled with a semi-Lagrangian (SL) treatment of advection (faster time-to-solution), called SL-DG [103, 102], thus departing from existing spectral element and DG models, which are nodal and Eulerian.

The modal version of the DG technique has the advantage of facilitating the introduction of a p-adaptivity strategy, i.e. being able to adapt the resolution by locally changing the polynomial order used to represent the solution, without affecting the mesh, which may remain constant. Even if this adaptive feature may not be of central interest for the use of SL-DG in the solution of the Euler equations, it can be disruptive if applied to a multi-tracer transport problem, as it has the potential of providing a completely independent resolution for each tracer when interpolating at the departure point of the mesh points (which are the same for all the tracers).

Currently under development at ECMWF, this novel SL-DG solver is going to be coupled with the *Atlas* library to exploit its mesh generation and partition capabilities, to facilitate intercomparisons with the existing approaches from IFS-ST and IFS-FVM to be run on equivalent resolution grids, and to ease the development of performance-portable code.

Constructing such SL-DG methods that scale and that preserve the conservation properties of DG methods represents a scientific challenge in the Scalability Programme.

Additional development efforts are required to provide suitable tangent-linear and adjoint models for the alternative algorithms, as well as programming models employed in the future. Given the trend towards further abstraction and modularisation, it is hoped that such tasks should become easier. Providing an efficient tangent-linear model is essential for effective 4D-Var data assimilation and requires increased attention in the coming years.

The atmospheric composition configuration, COMPO, (aerosols, greenhouse and reactive gases) of the IFS, which is developed and maintained by CAMS, aims to be fully aligned with the NWP configurations. COMPO should therefore benefit from the scalability and software infrastructure efforts being developed for the IFS. Obviously, mass conservation is important for COMPO so that alternative flux-form based, monotone advection schemes such as MPDATA and additional cost-effective solutions to run the IFS on multiple grids are an important research prospect for CAMS.

IFS model performance: Most of the information on the present IFS model performance has been presented in Section 1.2. Here we highlight additional aspects of coupling of the atmospheric component to the land surface, waves and ocean, and atmospheric composition.

Different Earth-system model components operate on different grids, with different numerical algorithms, using different temporal and spatial discretisation schemes (see Fig. 23). This implies spatial and temporal interpolation in the sequencing of information between the Earth-system model component models, and this may impose scalability constraints from one component onto another. While the individual, coupled components have fewer degrees of freedom than the atmosphere they may only be executed over sea points or only over land points, or may be completely independent from neighbouring points by design. This can still become a bottleneck if such a component is executed on the same number of processors as it runs out of parallelism. This is demonstrated by comparing the IFS model runs with the RAPS-18 benchmark in coupled and uncoupled mode. When run on 180 nodes of ECMWF's Cray

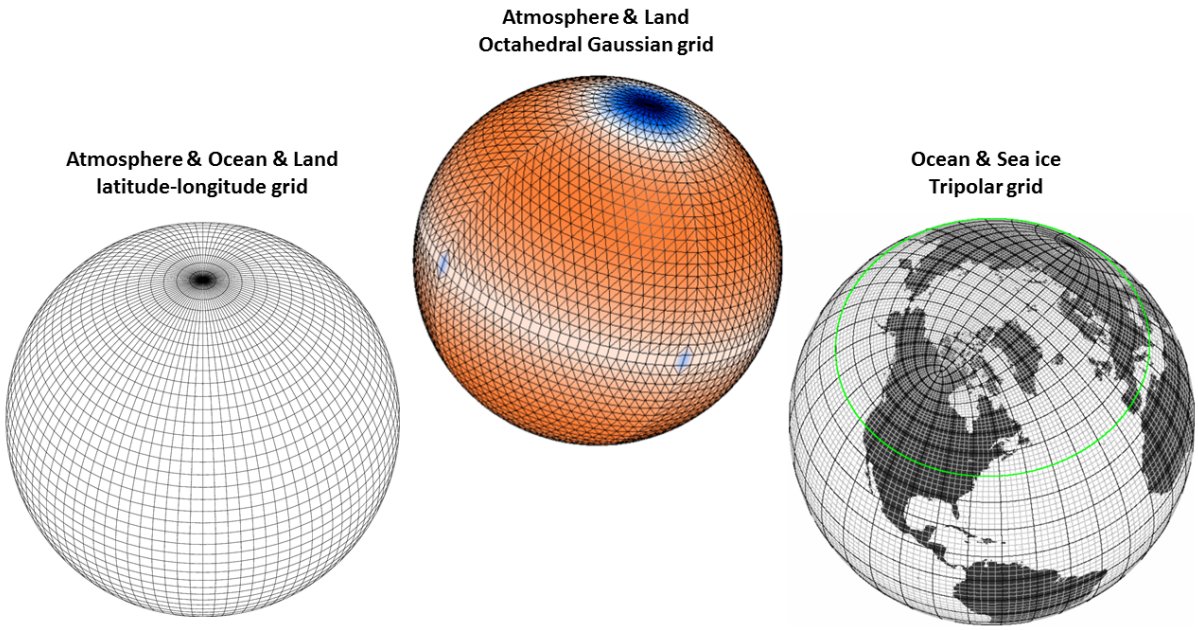


Figure 23: Different Earth-system model components such as waves, atmosphere, land, ocean and sea-ice are operating on different grids.

XC-40, the node-hour overhead of NEMO is about 26%, when run on 360 nodes the overhead becomes 63%. This justifies an investment in running such components concurrently as discussed in Section 2.3 and Section 4.2.

The effect of the workload imbalance for coupled components also becomes apparent when the resolution of the atmosphere is reduced as, for example, for monthly forecasts with a TCo399 (36 km) grid and the 1/4-degree NEMO set-up. The latter is fixed in all configurations due to the lack of similar-quality ocean initial conditions at different ocean resolutions.

In addition, algorithms and their implementation in community research models such as NEMO may not always receive the same attention on scalability bottlenecks as the in-house developed atmosphere component, because they are not routinely developed for time-critical service environments. However, this is changing as these components are now executed in the time-critical path at ECMWF and become increasingly visible. They equally form part of other Copernicus services (e.g. NEMO). This motivates research to explore the feasibility of mixed precision for other Earth-system components and further options as shown in Section 2.3, the consideration of operating with multiple grids and time steps (Section 4.2) and using surrogate models (Section 4.3).

The land-surface model CHTESSEL[71] uses a tiling approach for the different land-surface properties. The scheme is inherently parallel, closely aligned with the NPROMA-blocking of the whole IFS (and the physics in particular), and with all land points independent from each other. This facilitated the production of the 9 km ERA-5 - land dataset, with CHTESSEL among the most efficient land-surface models in the world.

The wave model, WAM, is solving the wave energy spectrum for a range of propagation directions and

[71] Carbon/Hydrology -Tiled ECMWF Scheme for Surface Exchanges over Land

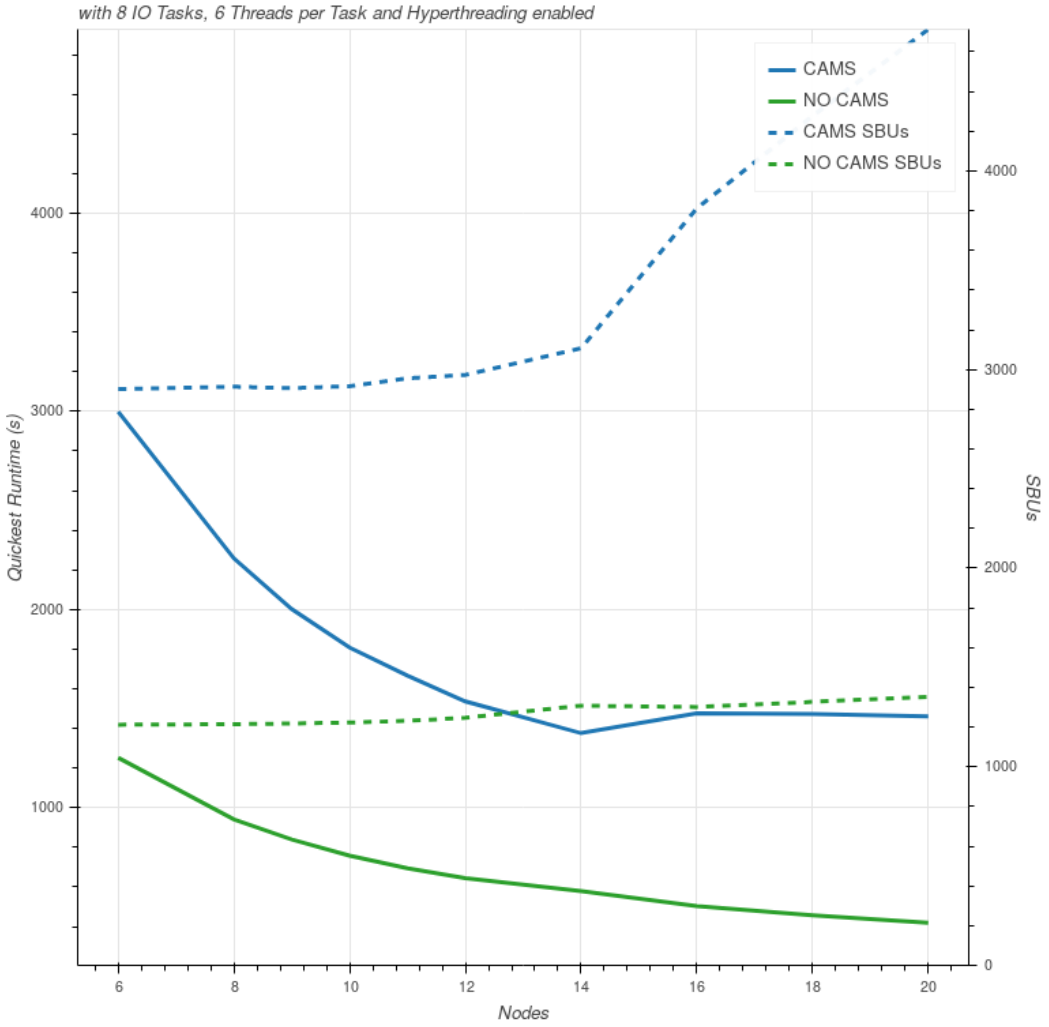


Figure 24: Scaling of run time (solid line) and computational cost (dashed line, standard billing units, SBU) of the IFS in the operational COMPO configuration (CAMS, blue) at TL511L60 and in NWP mode (no CAMS, green) at the same resolution. The IFS in COMPO configuration scales well until the I/O load on the I/O servers becomes too excessive. Because of the large number of tracers (ca. 80), the cost of I/O is enlarged. Increasing the number of I/O nodes increases the range of the good scalability for IFS-COMPO. IFS-COMPO is about 2.5 time more expensive than a NWP forecast at the same resolution.

frequencies. The advection is explicit, and limited to the domain containing sea surface waves, with compute intensive parts in the wave dissipation. Some of the bottlenecks arise due the transposition from the atmosphere to the wave model domain and the communication patterns within the wave model. There is a plan to use the wave model computations on the atmospheric grid, thus trading reduced communications with increased but parallel computations (together with the potential scientific benefit of increased resolution near the coasts).

With increasing expertise in alternative numerical methods and time stepping schemes suitable for other domains such as ocean, wave and sea-ice, the focus will also widen, e.g. towards the treatment of fast ocean surface modes, fractal coastlines, bathymetry, sea-ice rheologies, communication patterns in the wave and ocean models, which all present potential bottlenecks in a scalable future Earth-system model.

Adding aerosols and chemistry to the IFS produces an overhead factor of 2.5 - 3 in forecast mode and about a factor 2 for 4D-Var analysis experiments. Because of the substantially increased cost, the CAMS operational suite is run at a TL511 horizontal resolution and has only recently been upgraded from L60 to L137 with the implementation of cycle 46r1. The overhead of only including the greenhouse gases CO_2 and CH_4 is about 7% for a 10-day forecast run at a TCo1279 L137 resolution in the research configuration (60 nodes). On the other hand, using more sophisticated chemistry and aerosol schemes, which also include stratospheric chemistry or modal aerosol dynamics with about 120-150 tracers, would increase the cost by a factor 8-10 with respect to the configuration without composition.

Only about half of the overhead in forecast runs comes from the schemes that simulate chemical kinetics and aerosol dynamics. The remaining increase in cost comes from (i) the simulation of the transport (advection, diffusion and convection), (ii) global-norm calculations for global diagnostics and the tracer mass-fixer, and (iii) from I/O operations of the additionally simulated 56 chemistry and 14 aerosol tracers.

Systematic testing of the computational requirements of COMPO in research and operational mode has started. COMPO scales very similar as the IFS without composition but an increased number of I/O tasks is required for an efficient performance. Fig. 24 shows a comparison between the runtimes and the computational cost with and without COMPO enabled. The configuration of I/O tasks, threads per task and whether to use hyperthreads was chosen from those judged to be the best for COMPO. The scalability of the two types of runs are similar. The CAMS runs scale well until the I/O load on the I/O servers becomes too excessive, with the parallel efficiency reducing to 90% when going from 6 to 12 nodes. For comparison, the parallel efficiency for non-CAMS runs over the same range is 97 % and takes a factor of 2.5 less execution time.

2.3 Code adaptation and optimisation

In the first phase of the programme several options for performance enhancements were explored, both within the existing code infrastructure from using standards tools and compilers directives, and from trialling entirely new concepts. First steps towards an IFS component performance analysis have been taken, supported by external expertise.

2.3.1 Options within the existing code infrastructure

CPU assessment: One of the main vehicles for assessing code performance is the IFS RAPS benchmark, which has recently reached its 18th edition [93] in support of the present HPC-2020 procurement. In terms of complexity and real-application representativeness, this benchmark has greatly evolved and does now provide the framework to build and run both IFS and OOPS executables on various computer environments employing the available compilers and libraries. The benchmark comes with both single and double precision arithmetics. It provides input data and reference results for various model resolutions, so that benchmarks can be run using as little as a single core or hundreds of nodes for higher resolutions. It includes the PGen software that ingests the meteorological fields produced by the IFS and produces the disseminated forecast products. It also includes the *Kronos* software that creates a graph of the variety of tasks simulating a full schedule and generating background I/O loads for capacity benchmarks.

For establishing the expected performance range on present-day CPU processor architectures, RAPS18 has been implemented and run on the ECMWF Cray XC-40 Intel Xeon CPU E5-2695 (Broadwell) v4 CPU operating at 2.1 GHz clock-speed with 18 x 2 cores and 128 GiB memory per node. This was

compared with the AMD Naples EPYC 7601 32-core processor running at 2.56 GHz clock-speed with, 32 x 2 cores and 128 GiB per node. The chosen test was a 10-day forecast of IFS and WAM at TL399 L137 resolution, to be run in single precision, on a single node and without output.

Only in terms of throughput, a performance of 218 (199) forecast days per day was achieved with the Cray (Intel) compiler on the Intel processor, and 215 forecast days per day on the AMD processor. With future versions of the AMD processor, this may change as AMD Rome will have more cores and memory, and better memory bandwidth leading to an estimated 20% performance gain. Similar performance gains have been derived for ARM Cavium ThunderX2 32-core processors operating at 2.1 GHz clock-speed, however, not for the RAPS benchmarks but the NEMO ocean model and the Met Office's Unified Model [74]. If this test is representative, O(50%) gains can be expected from future CPUs, mostly from added cores, memory and memory bandwidth. However, if future CPUs will include extended vector register sizes, increased instructions per cycle, increased clock speeds, or contain high-bandwidth memory packages this gain estimate may be too conservative.

Performance analysis: In addition to the RAPS benchmark, a variety of tools are used on a regular basis to keep track of the IFS performance, and to help pin-point performance regression problems when they arise. A combination of in-house (DrHook, GSTATS, ecProf [36]), open-source (Paraver [18], Scalasca [19], Darshan [39], RubberDuck[59]), and commercial (Intel VTune [60], ARM MAP [4], CrayPAT [25]) tools are available. Using the in-house timing infrastructure, the collection and analysis of computational performance information from the IFS is now automated, making it easy for all developers to understand the cost of any new developments.

These tools are used routinely at ECMWF to identify opportunities for optimisations. As part of the research-to-operations process for each cycle, a full operational run is profiled to ensure that there are no unexpected performance differences compared to the previous cycle. This profiling can direct resources to optimise certain parts of the IFS. Such optimisations are part of business-as-usual work in the development of the IFS.

The IFS was profiled and analysed by the Barcelona Supercomputing Centre (BSC) using their Extrae and Paraver tools, as part of a Performance Optimisation and Productivity (POP) H2020 project [72] in collaboration with ESCAPE. BSC used a structured approach to characterise the performance of the IFS [107]. The findings from this study are shown in Tab. 3.

The BSC method gives a global efficiency factor which is the product of the parallel efficiency and the computation efficiency factors. The parallel efficiency metric is the product of the load balance, serialisation efficiency and communication efficiency. For small numbers of processes the load balance is the main cause of lower parallel efficiency. As the number of processes increases the communication efficiency becomes the dominant factor, which arises from the additional communications in the data transpositions in the spherical harmonics. The serialisation efficiency remains high, showing a high level of parallelism in the IFS.

The computation efficiency metric is the product of the IPC (Instructions per Cycle) scalability, the instruction count scalability and the frequency scalability. The IPC scalability drops slightly, while the number of instructions and frequency metrics remain high. Overall, the computation efficiency is higher than the parallel efficiency showing the need for improved scalability to deliver a higher parallel efficiency.

[72] Performance Optimisation and Productivity Centre of Excellence in HPC, https://pop-coe.eu

Table 3: BSC scaling study of the IFS at varying numbers of MPI processes.

MPI processes:	**48**	**96**	**192**	**288**	**384**
Parallel efficiency	0.865	0.843	0.760	0.744	0.707
Load balance	0.917	0.900	0.904	0.880	0.896
Serialisation efficiency	0.975	0.989	0.972	0.963	0.956
Communication efficiency	0.967	0.948	0.866	0.878	0.826
Computation efficiency	1.000	0.966	0.932	0.856	0.843
IPC scalability	1.000	0.974	0.955	0.896	0.891
Instruction scalability	1.000	0.993	0.976	0.950	0.943
Frequency scalability	1.000	0.999	1.000	1.006	1.003
Global efficiency	0.865	0.815	0.709	0.637	0.596

Enhanced performance in the present code implementation: The performance analysis tools have also been employed to perform more in-depth examinations of the code base, in order to better understand performance characteristics and scalability limits of the current code. An example of this is the study of memory access behaviour as a function of NPROMA, which is the batch-size of vertical columns being processed together. The NPROMA data blocking structure is the main tuning mechanism built in to the IFS to allow performance to be optimised over a range of CPU architectures and sizes, be they vector or scalar.

Not all areas of the code benefit in the same way from NPROMA. At one end of the scale, the Fourier transforms in the spectral transform package have a memory copy that is solely there to generate contiguous views of data to be transformed. The presence of NPROMA is thus purely an overhead here. At the other end of the scale, complex code which requires many temporary arrays benefits greatly from small NPROMA values, which minimise memory footprint and allow effective use of modern cache hierarchies.

Through a formal bilateral research collaboration with Intel, ECMWF has succeeded to implement a number of generic and Intel-compiler specific performance enhancements with the existing IFS. One example is modifications and bug corrections applied to the WAM code and allowing it to run with the Intel compiler. WAM is a mixture of Fortran-77 and 90 code where the lack of interface blocks caused not only illegal code generation but also triggered extra memory copies of some array arguments upon subroutine calls. WAM still suffers from non-ideal NPROMA-blocking (see above), which was identified by Intel. Other examples are that the IFS has been made bit-reproducible with the Intel compiler when linked with Intel's Mathematical Kernel Library (MKL), that SVML[73] calls (e.g. vector exponent) now work when floating-point exceptions are enabled like in the IFS, that the FFTW[74] algorithm stopped causing occasional floating-point exceptions through the MKL when running in AVX-512 mode, and through vectorisation improvements via SVML. This has improved the IFS performance by more than 10% and NEMO Performance through multi-threading improvements by more than 20% when using smaller-sized research configurations.

The ESCAPE project allowed performance improvements by breaking down the complexity of weather and climate models into verifiable mini-applications, NWP and climate dwarfs (described in more detail below, and Fig. 30), that can be shared and prototyped much more easily [82]. Bull worked as part

[73] Short Vector Math Library
[74] Fastest Fourier Transform in the West

of the ESCAPE project on optimising the MPI communication of the spectral transform method. The largest potential for optimisations was found to be in the preparation phase of point-to-point communications. During the preparation phase the sender side gathers the local data into a contiguous buffer (Pack operation) and hands it off to the MPI library. On the receiver side, data is then scattered from a contiguous user buffer to its correct location (Unpack operation). Pack and Unpack are nearly inevitable with scattered data because RDMA [75] with no gather-scatter operations are known to be often less effective, notably due to the memory pinning latency [111]. It also means that sender and receiver must exchange their memory layout as they may differ. The performance improvement came from reordering the loops for both pack and unpack following the memory layout of the scattered buffer. This optimisation decreased the number of tests (i.e. copy or not copy) and avoids scanning memory multiple times, which was unnecessary. This optimisation achieved a global performance gain on the spectral transforms of about 20%. More details about this work can be found in [82] and [28].

The collaboration with Intel and Bull, but also HPC centres such as BSC[76], shows the importance of ingesting vendor expertise with good knowledge of the IFS in a continuous way, and it demonstrates the impact complex applications like the IFS can have on compiler improvement. Beyond performance improvement, this process also helps to shorten spin-up time in the preparation of benchmarks (like RAPS) for procurements.

Mixed precision arithmetic: The use of reduced or mixed precision has been investigated for the IFS to improve numerical performance on existing and future supercomputers. Conventional weather and climate models are almost exclusively using double precision with 64 bits to represent real numbers. If precision is reduced to single precision with 32 bits per real number, numerical performance can improve significantly. However, to reduce numerical precision in a weather model such as the IFS is non-trivial due to the complexity of the model, the chaotic dynamics where small changes can cause unpredictable feedbacks, and the complicated interactions between different model components at different time-scales. Often, rather trivial operations like a division by a small number can cause problems and need to be expressed in a different way to work with reduced precision.

Still, an approach to change the working precision to 32 bits for the entire atmosphere and wave model made substantial progress over the past years and a single precision version of IFS is now prepared for use in operational forecasts [106] [34]. While some areas of the model code need to be maintained in double precision, such as the calculation of Legendre polynomials in model initialisation, the vast majority of parameters can be switched to single precision. Progress has also been made when exploring single precision in the ocean component NEMO [99]. A single-precision version of NEMO is planned to be implemented and tested within IFS for the next major ocean model upgrade with NEMO4 (see Section 6).

For model cycle 46r1, single-precision simulations show only very small differences in deterministic forecast scores and are neutral in ensemble simulations (see Fig. 25 and Fig. 26). The precision reduction causes the error in mass conservation to increase compared with double precision simulations but a mass fixer, which is both simple and cheap, can be applied to mitigate problems.

The use of single precision allows for a reduction of the atmospheric model run time by approximately 40% for forecast simulations. In general, the performance increase for single precision is caused by the cut of memory and cache requirements by a factor of two, the reduced data volume for MPI communication, and a speed-up for processing by a factor of 2 in areas of the code that are vectorised. However,

[75] Remote Direct Memory Access
[76] Barcelona Supercomputing Center

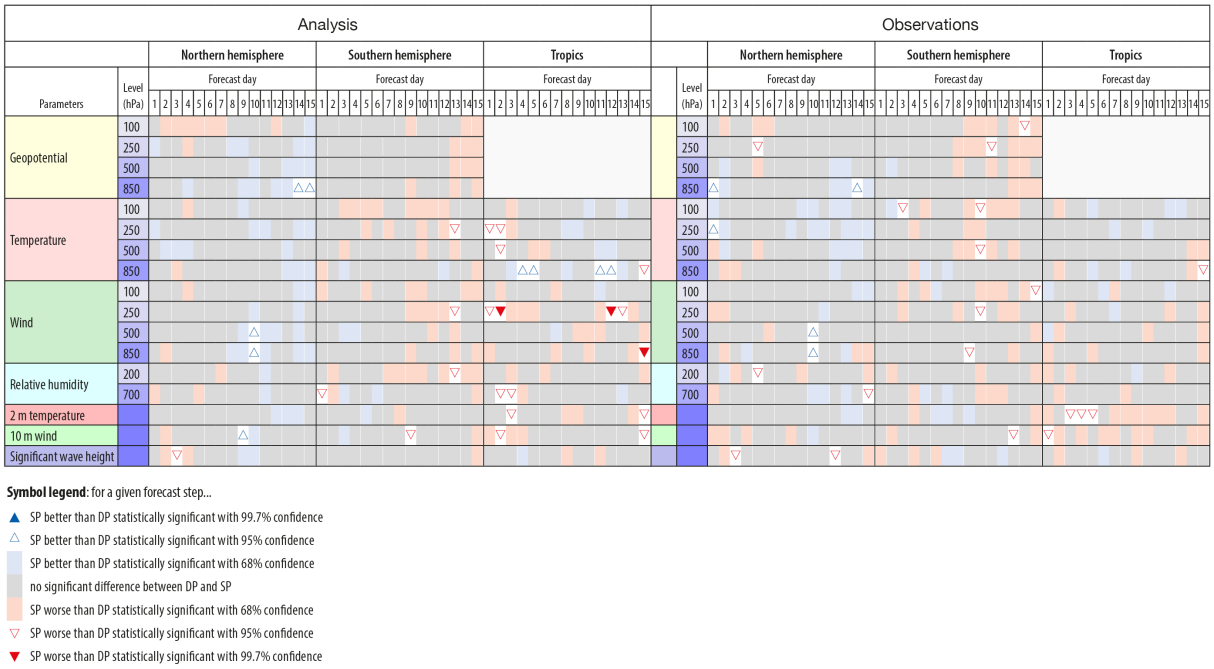


Figure 25: Ensemble score card for single precision experiments in comparison to double precision experiments for model cycle 46r1 showing the difference in skill (as fair continuous ranked probability skill score) between single and double precision for selected parameters and levels, verified with both analyses and observations.

the performance increase can vary for different hardware and compiler choices. A detailed performance evaluation of single and double-precision simulations on ECMWF's Cray supercomputer in collaboration with BSC revealed that the relative cost reduction from using single precision is different for the different components of the IFS. Overall, most benefits were generated by a smaller number of cache misses and improved vectorisation for single precision simulations [30].

The operational implementation of single precision for ENS and REF is planned for model cycle 48r1 after the migration of the HPC system to Bologna.

The use of reduced numerical precision was also studied in model output as part of the ESiWACE project ([33]). In particular, a compression technique for ensemble model output was suggested that would use similarities between ensemble members in ensemble forecasts to allow for a reduction of the number of bits in GRIB model output while keeping the maximal level of information. The method automatically generated output with high precision at the beginning of forecasts when ensemble members are more similar, to provide sufficient distinction, and decreased precision with increasing ensemble spread over time. To keep precision high for predictable situations and low elsewhere appears to be a useful approach to optimise data storage in weather forecasts [24].

Concurrent execution of model components: The approach of co-models was investigated as part of ESiWACE [80]. The idea is to run a number of co-models (or model components) concurrently within the IFS in order to break up potential sequential dependencies for the benefit of parallel execution. To allow for a concurrent integration, it is necessary for all co-models to run in a lagged mode using only data from an earlier time-step as input. Fig. 27 shows a schematic for the co-model implementation that was used in IFS where 13 threads have been assigned to the model core, 3 threads to the radiation scheme

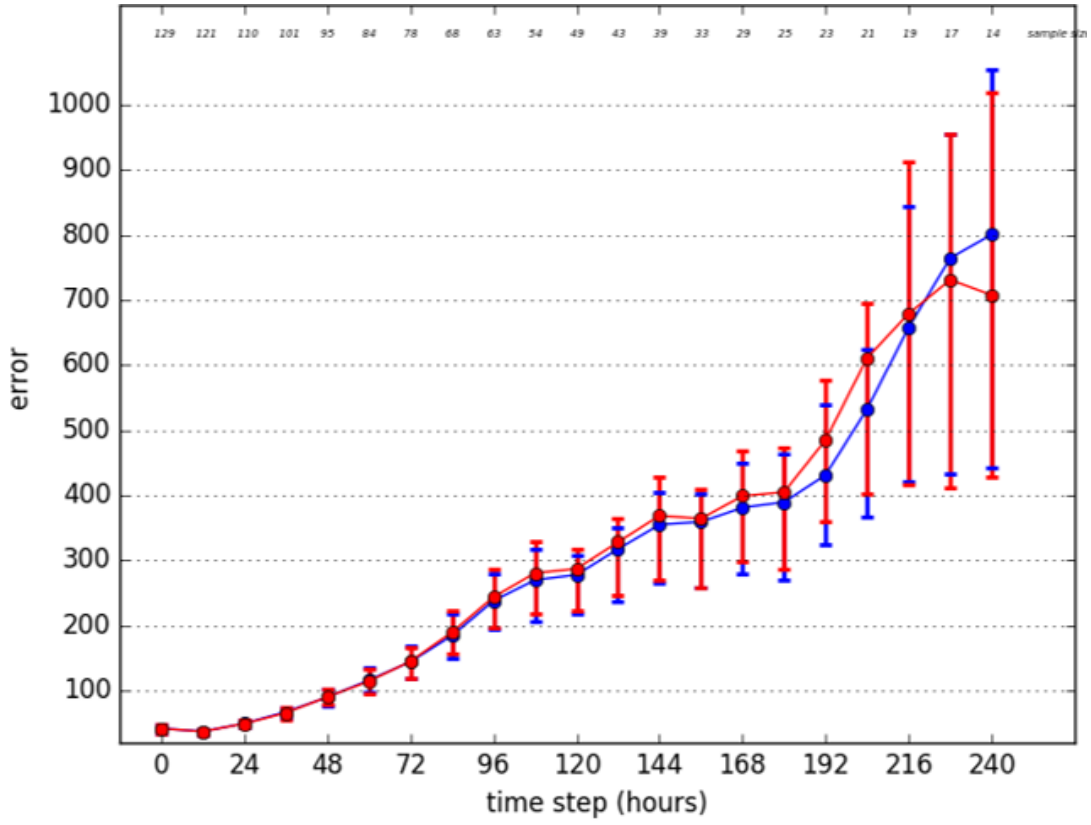


Figure 26: Tropical cyclone core mean position errors in km for double precision forecasts (red) and single precision forecasts (blue) at TCo1279 (9 km) horizontal resolution with 137 vertical levels. The sample contains all tropical cyclones that occurred across all basins during the period 18 August 2017 - 30 September 2017.

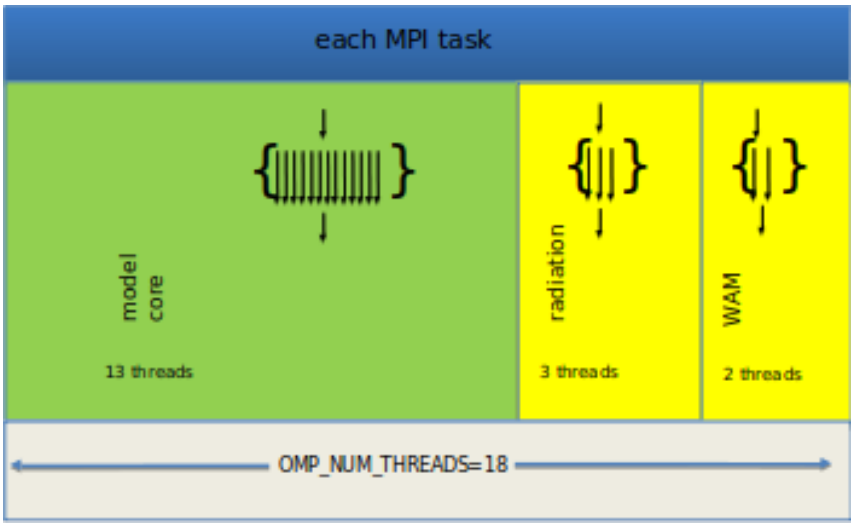


Figure 27: IFS implementation with radiation and WAM component co-models within shared memory space (radiation). The same method can be applied for distributed memory (wave model).

and 2 threads to the wave model. The exact number of threads to assign to each co-model is determined by trial and error testing with different combinations identifying the optimal usage of memory.

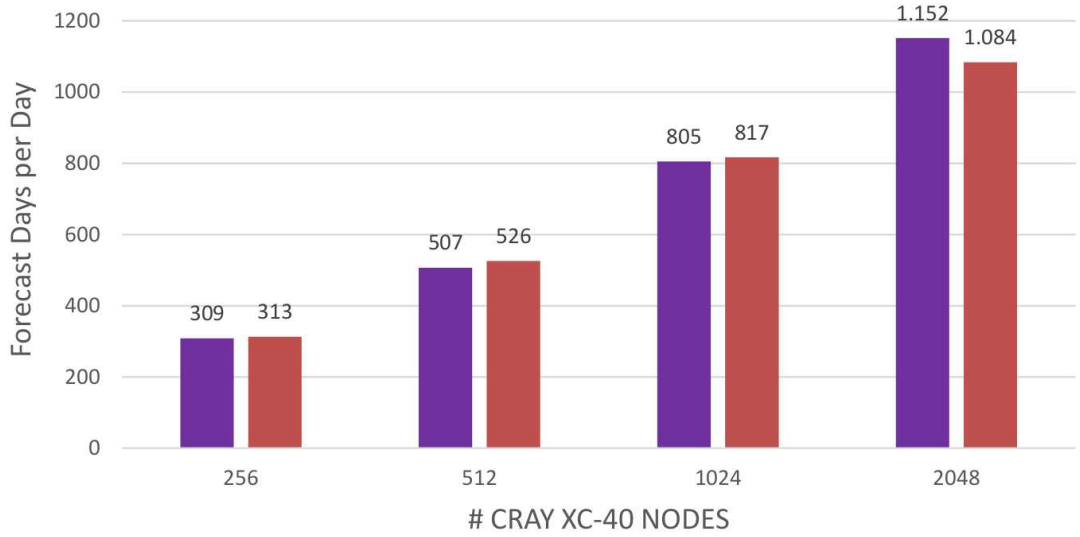


Figure 28: TCo1279 (9 km horizontal resolution) IFS forecast model throughput for different node allocations comparing control (violet) and co-model (orange) parallelising the execution of both radiation and wave model (including the communication fix in the wave model).

Fig. 28 shows the performance of an IFS 9 km (TCo1279) forecast model comparing the performance of control and co-model runs on one of ECMWF's CRAY XC-40 clusters. Performance is measured in forecast days per day. In the initial tests, the performance of the co-model configuration was better than the control, and the efficiency gain increased with increasing node allocations to up to 23% for 2048 nodes. However, this gain also included improvements in MPI communication in WAM. It was also found that the best performance for co-model runs occurred when only 1 or 2 threads were assigned to the wave model, and the remaining threads given to the model core and radiation scheme. This triggered a more detailed investigation of the wave model that identified a problem with MPI communication within the wave model. After re-coding the problematic communication, the parallelisation of the wave model was significantly improved causing the sequential integration of the IFS to look much better in comparison to the co-model version. For the fixed version of IFS, we observe that there is a marginal performance improvement with co-models up to 1024 nodes and a reduction in performance at 2048 XC-40 nodes. Nevertheless, the approach showed promise, and will be revisited with a more flexible implementation during the second phase of the Scalability Programme (see section 4.2.4).

Overlapping computation and communication: The communication between different model components and the limited scope for scalability for some components, such as the two-dimensional sea-ice and barotropic mode components in ocean models [64], can cause severe bottlenecks when the number of nodes is increased. Techniques of co-models or concurrency, that either overlap computation and communication or calculate different model components in parallel to decrease the need to scale the entire model to all compute nodes, are therefore promising for applications in Earth-system models.

In the short- and medium-term, overlapping computations and communications has been shown to be a promising approach to hide some of the communication costs present in the existing semi-Lagrangian transport and the spectral transforms [81]. Preliminary work in this direction was performed within the

CRESTA and EPiGRAM[77] projects and explored the use of Fortran2008 co-arrays and the GASPI/GPI-2 one-sided API[78]. Two points of contention were targeted in this work. Firstly, threads executing in the context of OpenMP parallel regions were allowed to take part in communication, thus allowing for a larger degree of overlap and parallelism compared to the existing approach where all communications were funnelled through the master thread. Secondly, one-sided communication was used to implement on-demand halos for the advection process, leading to smaller amounts of data transferred. Early results on the Cray XK-7 (DoE Titan) machine were promising as illustrated in Fig. 29. The use of Fortran2008 co-arrays led to a speed-up of up to 60% at 5 km horizontal resolution and at a scale of more than 220,000 processing cores.

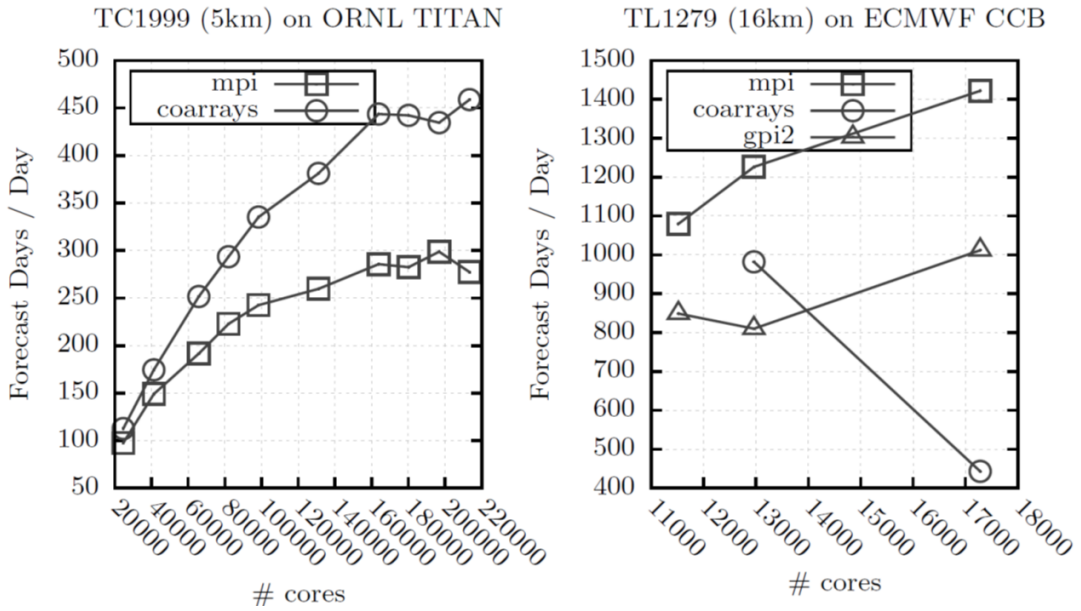


Figure 29: IFS model scaling runs on Cray XK-7 Titan (left) at 5 km horizontal resolution (TC1999L137) comparing baseline (mpi) and Fortran2008 co-arrays (coarrays) implementations and on the ECMWF Cray XC-40 (right) at 16 km horizontal resolution (TL1279L137) comparing baseline (mpi), Fortran2008 co-arrays (coarrays) and GASPI/GPI2 (gpi2) implementations.

However, the same co-array implementation, along with a version based on the GASPI/GPI-2 one-sided API obtained significantly worse performance on the ECMWF Cray XC-40 machine at 16 km horizontal resolution and on up to 18,000 processing cores. This disparity in performance between the runs on Titan's Cray XK-7 and ECMWF's Cray XC-40 were attributed to inefficiencies and bugs introduced in later versions of the Cray's compiler and DMAPP library on which both co-arrays and GPI-2 implementations were based and which were only discovered at a later stage.

An attempt to restart this work is currently underway within the scope of the EPiGRAM-HS project. The goal is to explore and assess the additional functionalities available in the MPI 3.0 standard, such as non-blocking collectives and improvements to one-sided and shared-memory communications, combined with the re-evaluation of the previous Fortran2008 co-arrays and GPI-2 implementations. It is hoped that runtime systems and associated libraries have matured enough in the meantime to deliver consistent performance improvements similar to the ones first observed on the DOE's Titan machine.

[77] Preparing Parallel Programming Models for Exascale
[78] Application Programming Interface

2.3.2 *Options using novel methodologies*

Processor technology: The rapid diversification of HPC hardware over the past decade has meant that incremental tweaking of existing software and approaches is appearing increasingly inadequate for achieving long-term performance goals. The investigation of what novel methodologies can bring has therefore been of continued interest to the Scalability Programme.

The ESCAPE project introduced the concept of dwarfs to the weather and climate community. The idea is based on the seven dwarfs of algorithms for high-end simulation in the physical sciences introduced by Phillip Colella in 2004 [23]. These were later extended to the 13 Berkeley dwarfs [5, 6] which are meant to represent characteristic patterns of computation and communication.

Following this idea, ESCAPE categorises key algorithmic motifs specific to weather prediction and climate models and identify their specific computational and communication patterns, which in return are crucial for the entire model performance (Figure 30). The dwarfs thus represent domain specific mini-applications [75] which include direct input from the domain scientist, documentation, timers for profiling purposes as well as error estimates for verification purposes. In this way the dwarfs facilitate communication of weather domain specific knowledge and algorithmic motifs with specialists in other domains. Different implementations of the dwarfs can be used as a first step towards optimising and adapting weather prediction and climate models to new hardware architectures and to benchmark current and future supercomputers with these simple but relevant applications. Identifying these key algorithms also allows better collaboration between operational weather prediction centres, hardware vendors and academia.

The concept of dwarfs is different from the existing separation of weather and climate models into different model components, such as atmosphere, land-surface and ocean for which separate dynamical core and physical parametrization packages already exist. Instead, dwarfs define a runnable and more manageable sub-component in a hierarchy of model complexity for specific targets such as adaptation to GPUs, exploring alternative programming models, and developing performance portable domain specific languages. But dwarfs can also be used by domain scientists for developing alternative algorithms, even across model components.

Having these ESCAPE dwarfs allowed targeted adaptation of specific algorithmic motifs to different computer hardware, and in particular accelerators such as GPUs that benefit from reduced precision use. Notable examples are the finite-volume transport algorithm MPDATA, the cloud microphysics, and the spectral transforms.

For the GPU version of the spectral transform dwarf, the prototype is restructured to: allow the grid-based parallelism to be fully exposed to the GPU in a flexible manner; ensure that memory coalescing[79] is achieved; and optimise data management. The performance on a single GPU is dominated by the cost of matrix multiplication (DGEMM) and FFT. For the matrix multiplications all of the matrices are padded with zeros to the same size. This allows to perform all matrix multiplications in the Legendre transform with a single batched call of the cuBlasDgemm library. This step increases the overall number of floating point operations by almost a factor 10 but still improves the overall performance (Figure 31). The FFTs are performed with the cuFFT library, batching over the vertical dimension. Multiple calls are still needed for each latitude since FFTs cannot be padded in a similar way to matrix multiplications. Therefore the implementation remains suboptimal. There would be scope for further improvements if a FFT batching interface supporting differing sizes were to become available.

[79] Coalescing memory access patterns is used to mean making sure that threads run simultaneously, try to access memory that is nearby.

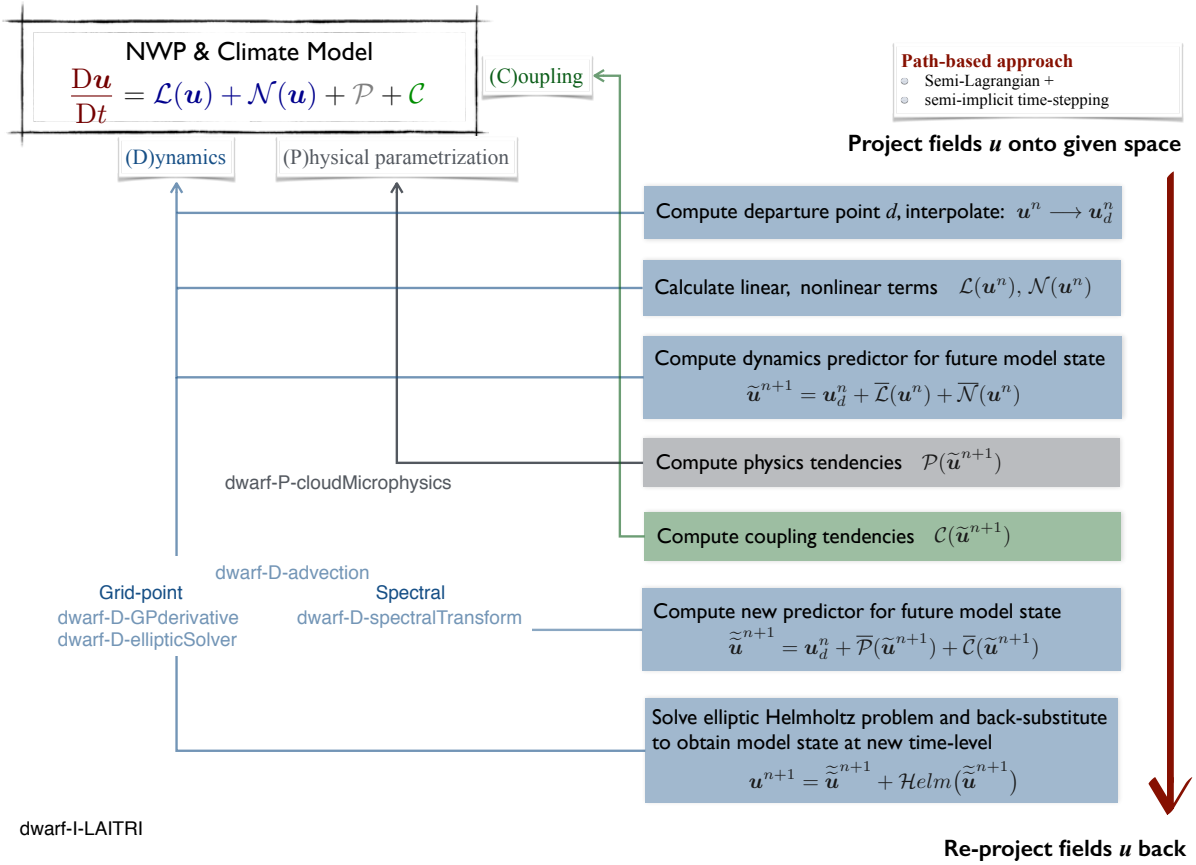


Figure 30: Illustration of the idea behind the ESCAPE dwarfs. The dwarfs are identified from the top-level building blocks of the model which are executed in each time-step.

More work has been done on restructuring the code which is described in detail in [82]. The restructured algorithm achieves an overall speedup factor of 23x compared to the initial version which also used cuBlas and cuFFT but followed the CPU version more closely. Matrix multiplication performance is higher than the overall performance (in flops) and the operational intensity is increased into the compute-bound regime. Note that matrix multiplication is associated with $O(N^3)$ computational complexity for $O(N^2)$ memory accesses. The extra padding operations lead to larger N and therefore also to increased operational intensity.

When running a single application across multiple GPUs, it is necessary to transfer data between the distinct memory spaces. Traditionally, such transfers needed to be realised via host memory and required the participation of the host CPU. Not only did this introduce additional latency, but also limited the overall bandwidth to the bandwidth offered by the PCIe bus connecting CPU and GPUs. However, modern MPI implementations are CUDA-aware. This means that pointers to GPU memory can be passed directly into the MPI calls, avoiding unnecessary transfers (both in the application and in the underlying MPI implementation). This is particularly useful when using a server that features high-bandwidth NVLink connections between GPUs, in which case CUDA-aware MPI will use these links automatically. Moving our dwarf to CUDA-aware MPI produced a speedup factor of 12 (Figure 32). However, even with this optimisation the all-to-all operations remained inefficient because communication between different GPUs was not exchanged concurrently. Perfect overlap was achieved by implementing an optimised

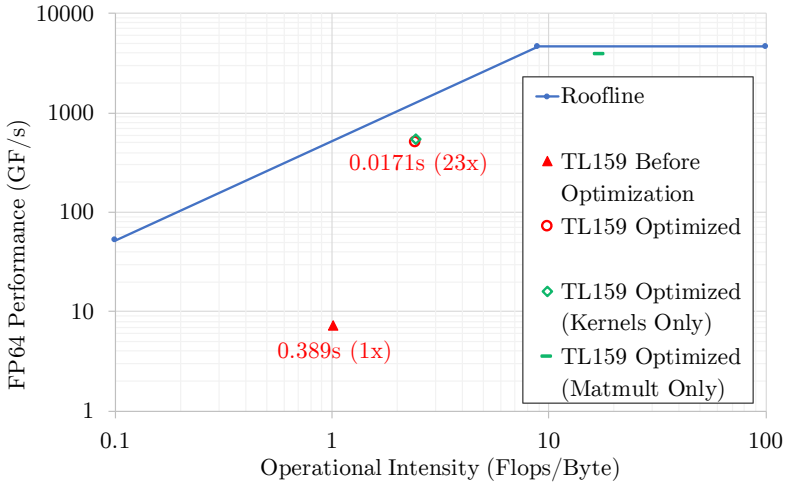


Figure 31: Roofline plot for the spectral transform dwarf at 125km resolution ($N_T = 159$) on the NVIDIA Tesla P100 GPU. The full time-step of the original prototype is represented by the solid red triangle. The corresponding time-step for the optimised prototype is represented by the red circle. Also included are partial results for kernels only (open green diamond atop the red circle) and matrix multiplication only (green dash). Each point is positioned in the plot according to its operational intensity: points under the sloping region of the roofline are limited by available memory bandwidth, and points under the horizontal region are limited by peak computational performance.

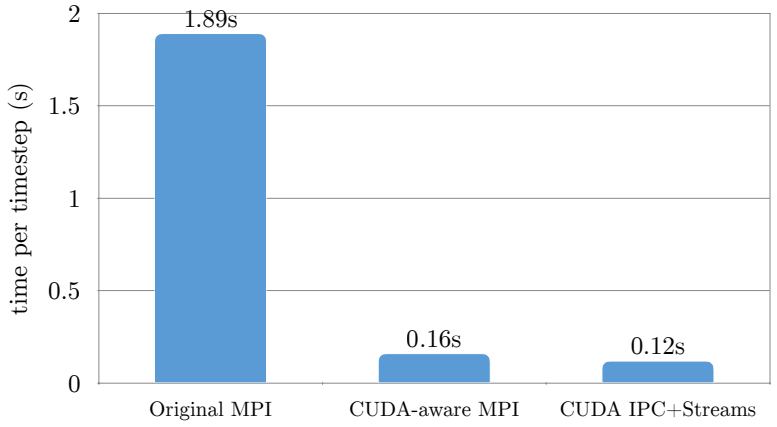


Figure 32: Computational performance of the spectral transform dwarf at 18km resolution ($N_T = 639$) on 4 NVIDIA V100 GPUs of the DGX1 with the original MPI implementation (left), CUDA-aware MPI communication (middle) and NVLink optimised communication (right). This resolution is currently used operationally for the members of the ensemble forecast at ECMWF.

version of the all-to-all communication phase directly in CUDA using the IPC[80] API. Using memory handles, rather than pointers, CUDA IPC allows to share memory spaces between multiple processes, thus allowing one GPU to directly access memory on another GPU. This allowed another speedup of about 30% (Figure 32).

Figure 33 compares the DGX-2 with NVSwitch and the DGX-1 for a spectral transform at 18km reso-

[80] Inter Process Communication

lution ($N_T = 639$). The NVIDIA Multi Process Service allows oversubscription of GPUs such that, e.g.

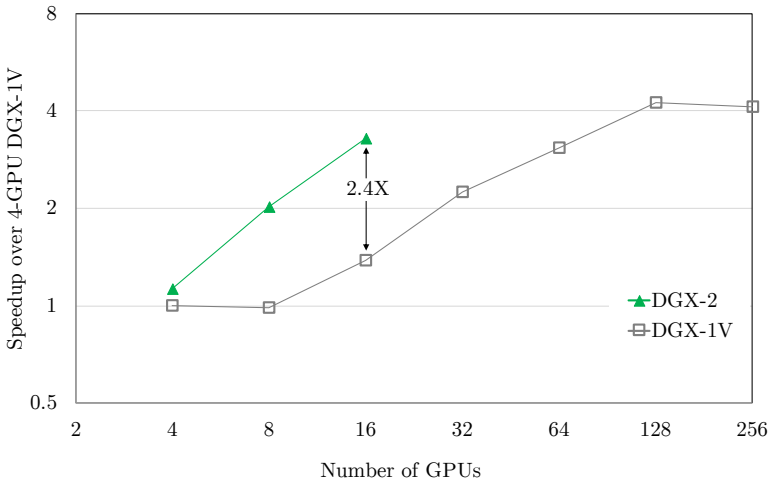


Figure 33: Computational performance of the spectral transform dwarf at 18km resolution ($N_T = 639$) on up to 16 GPUs on one DGX-2 and up to 32 DGX-1 servers connected with Infiniband. The DGX-1V uses MPI for ≥ 8 GPUs (due to lack of AlltoAll links), all others use CUDA IPC. DGX-2 results use pre-production hardware. The points were connected with lines for the purpose of improving readability.

the 8 GPU result on DGX-2 uses 16 MPI tasks across the 8 GPUs (i.e. 2 operating per GPU). Oversubscription can be beneficial to spread out load imbalance resulting from the grid decomposition and hide latencies.

The scaling on DGX-1V is limited with an increasing number of GPUs because some messages go through the lower-bandwidth PCIe and QPI links and/or Infiniband when scaling across multiple servers. On DGX-2 all 16 GPUs have full connectivity, i.e. the maximum peak bandwidth of 300 GB/s between each pair of GPUs is available. The performance scales well out to the full 16 GPUs on DGX-2 and is 2.4 times faster than the result obtained on DGX-1V. The speedup factor increase going from 4 to 16 GPUs on DGX-2 is 3.2, whereas the ideal speedup factor would be 4. Preliminary investigations reveal that this deviation from ideal scaling is not primarily due to communication overhead but due to load imbalance between the MPI tasks for a given spherical grid decomposition. This indicates that better scaling might be observed with a more balanced decomposition.

Figure 34 shows the spectral transform dwarf running on 11,520 GPUs, exploiting the Cuda DGEMM and FFT libraries as well as GPU direct communications via OpenACC instructions leading to an overall speed-up of up to a factor 23.7 compared to using only the Power9 CPUs on Summit on the same number of nodes. Particular accelerated systems such as NVIDIA's DGX1 and DGX2 and the arrangement on Summit offer high-speed connected multi-GPU arrangements that could potentially fit an entire ensemble member and therefore reduce the penalising communication aspects of the spectral transform method. The optimisations necessary to achieve this level of performance are described in detail in [82] and [28].

Optalysys investigated as part of the ESCAPE project an optical implementation of the spectral transform dwarf (biFFT for limited area models as well as spherical harmonics for global models). The fundamental idea behind optical processors is to encode information into a laser beam by adjusting the magnitude and phase in each point of the beam. This information becomes the Fourier transform of the initial information in the focal plane of a lens. The information can be encoded into the optical beam by using

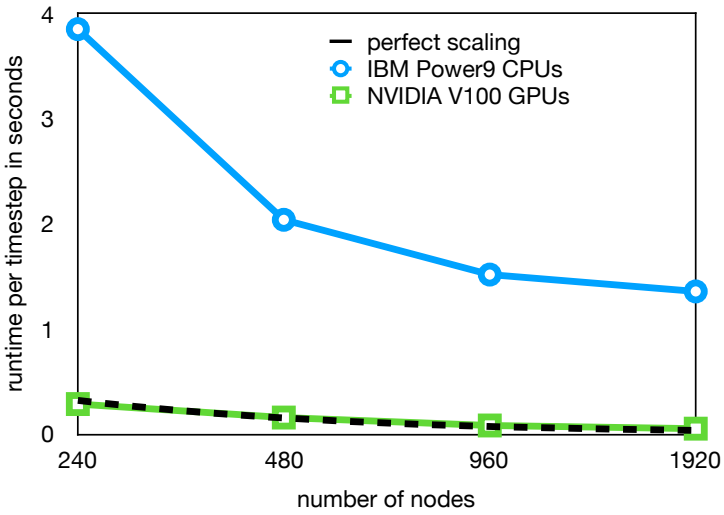


Figure 34: Scaling of spherical harmonics dwarf across multiple Summit nodes, using a hybrid OpenMP/OpenACC/MPI configuration, using GPUdirect for MPI_alltoallv and CudaDGEMM/FFT libraries across multiple nodes. Each node uses 6 MPI tasks and each MPI task uses one V100 GPU. The largest simulation therefore corresponds to 11,520 GPUs. Each Summit node contains two IBM POWER9 processors and six NVIDIA Volta V100 accelerators. Each physical core supports up to four hardware threads.

SLM[81]. The result of the Fourier transform can be recorded by placing a camera in the focal plane of a lens. This approach was found to consume very little energy but is too slow with multiple bit precision to be useful for weather prediction applications. The limiting factor is currently the frequency of the greyscale SLM. Optical processors are much better suited for applications where binary precision is sufficient [82].

Similar performance improvements on GPUs have been achieved for the MPDATA advection dwarf [28] and for the radiation dwarf [88]. These optimisations are hardware specific and will be difficult to maintain on upcoming new architectures. As a strategy towards performance portability and sustainability we have worked in the ESCAPE project on domain specific languages through the GridTools framework (see Section 4.2). The main goal of the GridTools library is to provide a solution for weather and climate models to run one code base on many different architectures (portability) and achieve good performance (performance portability). However, the main operational product of GridTools so far focused on solutions for latitude-longitude grid models like COSMO. The work developed in the ESCAPE project aimed at extending the DSL support for irregular grids and the efficient generation of back-ends for multiple architectures.

The DSL developments have been used to implement a portable version of the MPDATA dwarf. The DSL version hides such details as the nested loops and the OpenACC directives used to specify properties of the GPU kernel and data layouts of the FORTRAN arrays. Furthermore, the DSL allows to compose several of these operators together, which is used by the library to apply advanced performance optimisations like loop fusion or software managed caches. Comparing the FORTRAN OpenACC kernel with the DSL version gives us a speedup factor of 2.1 for the DSL version. This speedup could also be achieved by hand-tuned optimisation. The DSL prevents the repeated manual effort of tuning the code for multiple architectures. At the same time the DSL allows to perform optimisations which would

[81] Spatial Light Modulators

otherwise make the code unreadable. More details about this work including code examples on how to use the new back-end to GridTools can be found in [83]. More work on sustainability through domain specific languages will follow through the ESCAPE-2 project.

Dataflow computing: Another computing paradigm that is under continued investigation is the use of dataflow computing for meteorological codes. For problems where it can be applied, dataflow computing, using FPGAs[82] or in some cases ASICs[83], has the potential to achieve breakthrough reductions in energy per unit of work, or energy to solution.This energy efficiency stems from the fact that dataflow computing reduces the memory traffic required to perform a set of operations by effectively computing at the location of the data. Since the improvement in energy efficiency of floating point operations has been far larger than that of memory transfers over recent years, a memory transaction now consumes vastly more energy than a double precision floating point operation, see Figure 35. Due to the use of pipeline parallelism, dataflow architectures, such as FPGAs, are still able to perform large numbers of operations simultaneously at competitive throughput but with far fewer memory or cache accesses and at vastly reduced clock-speeds, resulting in a significantly reduced power consumption.

However, FPGAs have been notoriously hard to programme. Although it is now possible, thanks to efforts from vendors, to target them from high-level languages such as C, OpenCL and MaxJ, the development of a commercially supported Fortran compiler for FPGAs appears improbable. Moreover, for a given algorithm, a C-based implementation targeting CPUs is unlikely to perform well on an FPGA, due to the fundamental differences between CPU and FPGA architectures [77]. ECMWF's participation in the EU-funded H2020 EuroEXA project has allowed the investigation of FPGA usage for microphysics parametrizations in the IFS. Alongside the ongoing development of tools targeted at automatically translating Fortran to FPGA-compatible code, a proof-of-concept manual port of single-precision CloudSC has been undertaken, yielding positive results. Preliminary testing on a Xilinx VU9 is showing a speed-up factor of 3.6 over a single-socket of Intel Haswell E5-2690 v3 CPU at a third of the dynamic power usage, as well a speed-up factor of 2.5 over a single socket of Intel Broadwell E5-2695 v4 at a fifth of the dynamic power usage, demonstrating a potential improvement of energy to solution by up to an order of magnitude [98]. This investigation of current FPGA usability for IFS code suggests that despite the significant barriers to adoption in a model code, they are a promising avenue for future cost-effective HPC systems, and increasing in-house knowledge of how to programme them is important.

Machine learning: 'Machine learning' was used to improve and develop models from the very beginning of numerical weather predictions, for example in the form of linear regression or a principle component analysis. Furthermore, data assimilation can, in general, be interpreted as machine learning. However, new methods of machine learning that have hardly been used in Earth-system science, such as deep neural networks or decision trees, have improved significantly in recent years and became much more powerful as super-computing hardware and the amount of data that is available for training ('big data') has evolved. The number of applications of machine learning is steadily growing and deep learning tools have recently shown an outstanding capability to finish difficult tasks much quicker and/or better than humans (for example when counting cars in images or playing the board game 'Go').

The new tools and new technologies will likely have a significant impact on the Scalability Programme and ECMWF in general on many different levels, including quality control and automated alarm systems, data assimilation and use of observations, numerical modelling, and the post-processing of forecast and

[82] Field-Programmable Gate Arrays
[83] Application-Specific Integrated Circuits

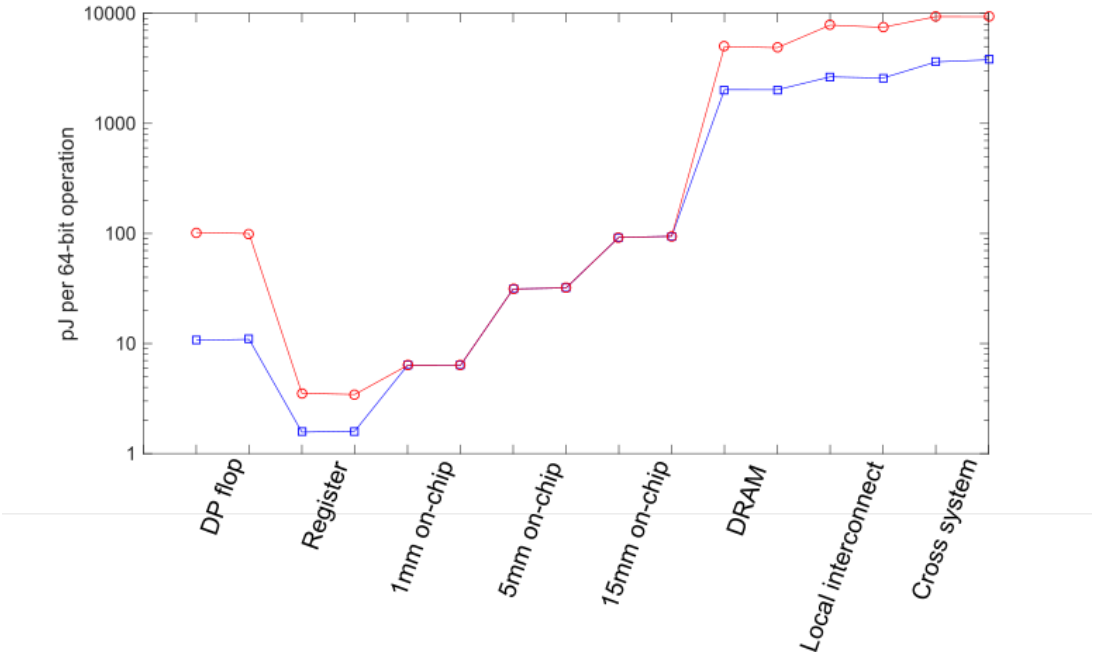


Figure 35: Energy consumed by different operations involved in HPC computing for 2008 (red) and 2018 (blue) hardware, respectively [63].

climate reanalysis outputs [31].

To approach new machine learning techniques for use in numerical weather predictions and to understand future challenges and fundamental design choices of deep learning tools for weather models, there has been some experimentation to learn the equations of motion of the atmosphere from climate re-analysis data [32]. This study retrieved global data for geopotential height at 500 hPa (z500) from ERA-5 re-analysis for 67,200 time-steps that are all one hour apart and mapped the data onto a coarse resolution grid. The resulting data set was used to learn the equations of motion for z500 by learning the update from one time-step to the next using deep neural networks. The resulting network could be combined with an explicit time-stepping scheme and used to generate weather forecasts into the future. The neural network models performed well in terms of forecast errors when compared to forecasts of the IFS model at a similar level of complexity with T_L21 resolution. The flow dynamics of the neural network forecasts also showed very realistic dynamics for the first couple of days and could hardly be separated from the ERA-5 time series by eye.

However, the study concluded that it is unclear how to increase model complexity of the neural network forecast model to obtain more fine-scale features typical for conventional forecasts with high resolution. The neural network approach will eventually face the same challenges when compared to the development of conventional models, such as the complexity of the Earth system with non-linear interactions between model components, scale interactions, exponential growth of errors in initial conditions, numerical instabilities and the discrete representation of model fields on the sphere, the treatment of conservation properties, model biases in long-term simulations, errors in observations, and insufficient data coverage of observations.

One of the most promising applications of new machine learning tools in weather and climate models is the emulation of existing model components of weather forecast models using neural networks and in

particular the emulation of parametrization schemes. If it would be possible to emulate a parametrization scheme with sufficient quality, there would no longer be a need to rewrite the original parametrization scheme with domain specific languages since it would be trivial to port the resulting neural networks to heterogeneous hardware using standard machine learning libraries such as TensorFlow or MXNet. The calculation of neural networks is also very likely much faster when compared to the original parametrization scheme due to the use of simple linear algebra and matrix-matrix multiplications, and a reduction of complexity. Finally, the approach may also facilitate the generation of tangent linear and adjoint versions of forecast model components that are required for 4D-Var data assimilation since the structure and algorithms of neural networks could be much easier to translate.

The idea to emulate IFS model components was realised successfully twenty years ago at ECMWF when components of the radiation scheme were emulated using relatively small neural networks with a single hidden layer [20]. In a collaboration with NVIDIA, ECMWF is now in the process to generate neural network emulators for the current radiation scheme of IFS for model simulations with 137 vertical levels. The neural network emulators for both short and long wave radiation are trained from the full sets of input and output fields of the three-hourly radiation scheme that were stored from a full year of T_L159 simulations. A number of different combinations of data pre- and post-processing and different network architectures have been tested and preliminary results show that model simulations that are using the neural network scheme instead of the original radiation scheme are indeed running stable and showing reasonable results in terms of forecast errors.

Preliminary estimates of computing cost suggest that the neural network scheme is at least a factor of 10 faster when running on NVIDIA Volta GPUs than the original radiation scheme on XC40 nodes at ECMWF. However, differences between the schemes in terms of model fidelity are still visible for single model simulations and a much more detailed evaluation of forecast quality and computational cost is currently performed to decide whether the quality and speed of the neural network emulators will be sufficient for forecast simulations.

Machine learning will also have a very strong impact on future developments of high-performance computing hardware due to the strong demand of matrix-vector calculations at high FLOP-rates but with low numerical precision (see the Google TPU or NVIDIA Tensor Core architectures). It has therefore been tested whether this technology can be used to accelerate IFS simulations. In particular, it was investigated in a collaboration with Oxford University whether half precision floating arithmetic could be used to perform the Legendre transformation with IFS which is generating a significant fraction of total cost for simulations at high resolution. If the field values are re-scaled and if the transformations with very low zonal wave-numbers are secured to higher precision, results are indeed very promising and suggest that NVIDIA Tensor Core deep learning accelerator may allow for a significant performance increase of the Legendre transformations in the future [54].

3 The first phase: Achievements

As shown in this section, the first phase of the Scalability Programme has been very successful at initiating a large-scale and concerted effort to assess the key bottlenecks in the present forecasting system taking into account the entire prediction workflow and creating a focus on both data handling and computing aspects. Only through this holistic approach could all levels of granularity - ranging from workflow to code elements - and technology - ranging from processors to clusters and file systems - be addressed.

The programme exploited existing means of performance optimisation without radically changing code and workflow structures, but it also launched advanced research into entirely new concepts of program-

ming and data-centric workflow management. Obviously, the new concepts require careful planning as their implementation will be intrusive. As the complementarity between efficiency oriented computational science developments and algorithmic research is becoming increasingly intertwined and science choices may be outweighed by performance choices, scientists need to be convinced that the future system will allow them to actually do more research more efficiently.

Hence the development of a data assimilation and modelling framework that is as open and configurable as possible is essential for achieving the best trade-off between science and computing performance, and for achieving acceptance. The lead-time for such developments is usually O(10 years) and needs to be started well in advance of the emergence of design constraints imposed by computing. Important elements of this development have already been achieved in the first phase of the programme.

In brief, the following list summarises the main achievements in Phase 1.

Performance assessment: The establishment of the status quo is important to understand and quantify the general performance of the forecasting system in the present configuration, and on the range of the available hardware architectures both inside and outside ECMWF. Part of the status-quo assessment is also a deeper analysis of shortcomings at which research and development can be directed.

- The performance of the spectral-transform model IFS-ST has been established on the available CPU architectures (Intel, IBM, ARM, AMD), both at ECMWF and the largest HPC infrastructures in the world: (1) The parallel efficiency of the present IFS is good on present node allocations but drops below 60% on very large node allocations. However, the IFS remains very efficient in terms of time to solution due to its long history of MPI/OpenMP optimisation, enhancements of spectral transforms and the large SLSI time steps, achieving more than 100 forecast days per day at 1.45 km resolution on allocations of 3,800 nodes on Summit. However, this time-to-solution performance can only be achieved with fast (and potentially expensive) interconnects between nodes. (2) Adding NEMO and I/O drastically reduces both scalability and efficiency, and WAM also becomes a significant cost driver at high resolutions with large compute node allocations. Atmospheric composition adds another significant cost factor depending on the number of variables involved in the advection of species. (3) The non-hydrostatic IFS-FVM dynamical core offers significantly better strong scaling and has the potential for a highly competitive time- and overall cost-to-solution on future architectures.

- Tests across individual CPU processor types offers performance speed-up of 20-50% from added cores, and more memory and memory bandwidth. This appears to be the maximum achievable gain on such architectures in the future if the code remains the same. Projections of computing cost with better-resolution, coupled model ensemble analyses and forecasts have been produced for the HPC2020 business case and resulted in growths factors of 12-15 for the next major upgrade - which can clearly not be compensated by the architecture alone.

- The main performance bottlenecks for computing are caused by the lack of arithmetic intensity. This is driven by insufficient memory bandwidth, excessive data communication on/off memory, sub-optimal memory access patterns from loop ordering and vectorisation, load imbalance, and data communication between nodes that is energy inefficient and causes wait times through synchronisation. Other bottlenecks include over-specification of numerical precision and unnecessary sequential dependencies. For data handling, the main bottlenecks are load imbalance across nodes, unnecessary data transfers across the memory hierarchy including disks, lack of parallelism of I/O servers, files-based data organisation and access, unnecessary tasks being handled within the critical path, and inefficient file systems.

- The investment in continuous observation pre-processing, continuous data assimilation, load balancing for the observation - model grid-point match-up, and running the single executables under OOPS has led to a performance in which (1) observation processing including observation operator cost has nearly become negligible and (2) efficiency and scalability of the data assimilation is mostly driven by that of the model and its tangent-linear and adjoint versions.
- These performance results differ significantly between the operational production and research experiment versions of the code, mostly due to the respective spatial resolution and the reduced compute node allocations for research experiments (tuned for capacity rather than time-to-solution). This leaves room for capability vs capacity optimisations.
- Projections of data volumes for the next 5 years indicate that there will be O(10) more observational and model output data that the workflows need to be prepared for now. Beyond 2025, these volumes will grow drastically through more satellite and IoT data, and kilometre-scale model simulations.
- Performance assessment and monitoring for both operations and research is now supported by a large number of internally and externally developed software tools that allow tracing of efficiency bottlenecks as well as detailed code optimisation.

Data handling and workflows: The growth of data volumes and diversity, both from observational input and model and product output, required to invest in key tools for managing fast and flexible data access and minimising data transfer across the memory hierarchy.

- A major development has been a completely new object-store for model output, that became the fifth version of the FDB. It features the full integration with the MARS ecosystem and allows for new workflow optimisations. It also supports novel I/O architectures such as NVRAM DIMMs as developed within the NextGenIO EU project. This technology demonstrated linear scalability for throughput and achieved roughly 10 times better performance over the current Lustre/HDD solution. FDB5 became already operational with model cycle 46r1.
- The research into novel methods of post-processing and generating products under the auspices of the Scalability Hermes project has led to critical insight into the successful development of the new product generation system. These included the development of a linear-algebra library abstraction which supports both CPUs and GPUs. Moreover, in collaboration with the ESCAPE project, a new serial implementation of the spectral transforms was developed to help accelerate product generation and minimise its memory footprint. These developments have already been successfully deployed into time-critical operations, and their combination has led to a factor 5 improvement in the generation of products.
- The development of the *Kronos* workflow simulation and benchmark generator software is entirely new and a major step towards realistic capacity benchmarking. *Kronos* has been used in the most recent HPC procurement to provide vendors, for the first time, with a full workflow representation including IFS model output and product generation. This has made the benchmarks much closer to the reality of running our operational system on the next HPC infrastructure.

Data assimilation and model development: The co-development of frameworks for flexible science choices and their implementation on various (heterogeneous) computer architectures is essential for trading off scientific and computing performance in the future. Model and data assimilation system are strongly inter-dependent and need generic infrastructure elements.

- The OOPS concept was developed initially to run and evaluate new variational data assimilation algorithms using simplified models. To extend this to the full IFS code required major code refactoring. It has been successfully demonstrated that the OOPS-IFS can match IFS in terms of accuracy of the analysis and complete the 4D-Var step significantly faster than the current framework (at present at least 20%). In addition the OOPS-NEMOVAR interface is in place and close to being capable of reproducing the current operational configuration.
- Inevitably, having the capability to run the IFS in two different setups has revealed deficiencies not only in the new configuration, which were fixed, but also in the existing framework. On several occasions when differences between OOPS-IFS and IFS were encountered, the issue was found to be in the IFS code. These IFS deficiencies have been corrected and improvements delivered to the current operational system, both in terms of accuracy and optimisations affecting overall performance. It is likely that further optimisations will be found.
- The new OOPS-IFS system is expected to be more robust and, after a transitional period when code developers get used to OOPS, should make code maintenance simpler by separating more clearly different functional units in the system.
- The OOPS code structure will be the basis for research and development in data assimilation over the next decade or more, allowing developments that should be able to ensure that ECMWF's strategy for data assimilation can be delivered. This is enabling research in aspects such as randomised Singular Vector Decomposition to improve the minimisation. It also means alternatives to 4D-Var, should this prove necessary, such as 4D-EnVar can now easily be evaluated if required.
- The successful *Atlas* development of a generalised and flexible framework for data structures for the IFS enabled substantial progress with alternative discretisations and alternative model developments. *Atlas* is already implemented in the MIR software package for use in the operational product generation and MARS interpolations.
- The successful weather and climate dwarf concept enabled rapid prototyping of alternative SL-advection formulations, of which some ideas fed into the operational model. Equally, the acceleration work on the spectral transforms was pioneered in the stand-alone dwarf, as well as ongoing work with the overlapping of computations and communications, and the cloud physics dwarf, the multiple grid option, the DG work, and first implementations of the conservative advection for example. Many of these developments are based on Atlas.
- The successful development of the IFS-FVM (also based on the *Atlas* data structures) has delivered an alternative non-hydrostatic dynamical core option with conservative advective transport operating on the same grid and with the same physical parametrizations as the operational IFS. The model adds to IFS a range of novel and complementary features that are expected to become particularly important towards emerging HPC architectures on the one hand, and high-resolution, convection-resolving, forecasts on the other.

Performance optimisation: Beyond the status-quo performance assessment, significant efforts were invested in exploring new sources of efficiency gains, both within the existing code on new processor technology as well as with entirely new concepts.

- Several algorithmic patterns, expressed as dwarfs, have been successfully ported to GPU, FPGA and even novel optical processors, including first explorations in the context of DSL tool-chains. On single processors, optimisation of dwarfs for GPUs show substantial performance improvements by factors over 20 for spectral transforms, MPDATA advection and radiation. Using a

DSL instead of a simple OpenACC implementation for MPDATA produced a speed-up factor of 2.1. Optimisation of selected dwarfs on CPUs only produced gains of 20%. The cloud scheme achieved a 2.5 times speed-up on FPGA and an estimated factor 10 in energy to solution.

- When run on multiple nodes and taking advantage of the fast NVSwitch interconnect on DGX2, a scalable and significant (x20) acceleration was achieved for the spectral transforms on 16 GPUs. Very good strong scaling efficiency was obtained when running many-fields spectral transforms at 2.5 km resolution on up to 11,520 GPUs on the supercomputer Summit, with a factor 24 better throughput than on IBM Power9 CPUs. Note that the IFS performance per node on Summit CPU is already by a factor 2 better than on Piz Daint CPUs, mainly because of having two CPU sockets in addition to the GPUs.

- Enhanced load balancing for the model-observation match-up in the screening procedure reduced the cost of the screening including the radiative transfer operator to negligible levels for the operational configurations. In lower resolution research experiments, the relative cost of the screening is still high as the same number of observations is being processed.

- The current IFS I/O server (initially developed at Météo-France) has been extended to be used for the ENS model runs and specifically for WAM, where it reduced the number of MPI tasks dedicated to I/O and introduced asynchronous model output, thus reducing the time-critical cost of the WAVE model.

- Three important performance enhancements have been tested using the existing code, namely (1) the use of mixed-precision arithmetic in the IFS for forecasts with an efficiency gain of about 40% (to be implemented with cycle 48r1), (2) the use of concurrent co-model execution for radiation and wave model (for both shared and distributed memory) with an efficiency gain of up to 15% at high resolutions (to be implemented as soon as possible), (3) and the use of overlapping communication and computation with existing and new programming standards. Efficiency gains of up to 60% were achieved on Cray XK-7, but remained much smaller on Cray XC-40, mostly due to compiler issues.

- The IFS has been ported to other CPU-type architectures using the RAPS benchmark, also in preparation for the present procurement. This helped (1) to ensure the code compiles and can run without much added effort during the procurement and (2) establishes an estimate of performance on Intel vs ARM vs AMD CPUs and associated compilers. Including the above stated efficiency gains, this leads to an estimate of the maximum achievable speed-up on current-technology CPUs with the (existing or moderately modified) IFS code infrastructure by a factor of 3, beyond which no significant acceleration is expected.

- Tests investigating the use of machine-learning for replacing costly IFS components (here radiation) with surrogate models produced efficiency gains of up to a factor of 10, which could reduce the overall atmosphere-model execution time by 20% if all parametrizations were replaced. Overall scientific performance and code reliability needs to be verified.

Collaboration and leadership: Hardly any of the above work could have been performed without the collaboration and funding framework that has been created since the foundation of the Scalability Programme. Apart from pooling resources and getting access to expertise that is not available at ECMWF, many of the past and future developments require Member State and weather/climate community buy-in as they are disruptive and need to serve more than a single prediction system.

- ECMWF established weather (and climate) prediction as one of the leading HPC and big-data handling applications in the European HPC ecosystem, and ECMWF as the leading entity for driving advanced research developments in this field. This is being extended to the field of machine learning at present.

- ECMWF supported the European Commission's strategic planning and funding programme preparation through membership in ETP4HPC and the formulation of their Strategic Research Agenda, which serves as input for the Commission's funding programmes.

- ECMWF acquired substantial external funding of about €1.5 million per year between 2015 and 2021 from more than 10 projects. This funding complements ECMWF core-funded staff previously allocated to the Scalability Programme.

- ECMWF established cross-departmental collaborations at the Centre and drew in a new generation of young computational scientists that will ensure the sustainability of the IFS and incorporate modern software design concepts for the entire prediction workflow.

- ECMWF founded a collaboration framework with weather and climate prediction centres, academia, HPC centres and industrial partners such as Intel, Bull, NVIDIA, Fujitsu, founded new collaborations outside Europe, in particular ONRL, NOAA ESRL[84], NCAR[85], and substantially raised the overall Scalability Programme theme at WMO level.

- The Scalability Pogramme provided templates for national strategic programmes at national hydrometeorological services and research organisations.

- Through the ExtremeEarth Flagship proposal, ECMWF created a visionary European science-technology collaboration that is likely to influence future weather and climate research infrastructures, and future service provision.

In the first phase, the limitations of the existing software infrastructure have been clearly established, and it is evident that a sustainable and well performing forecasting system requires several new and radically different developments. These could only be partly identified and tested in the first phase. An important input to this work is a more detailed performance analysis that delivers information on data communication bottlenecks across the memory hierarchy, the degree of achieved parallelism, load balancing and their dependence on architectures. A detailed performance analysis is also crucial for a quantitative assessment of how the specific performance of individual IFS components add up to the overall IFS performance, and how efficiency gains achieved for individual IFS components will translate into efficiency gains for the entire system.

A major outcome of the first phase is that, in addition, the main choices for future development have emerged and their transition to an operational implementation can now be planned. As Section 4 will lay out, the main two foci will be 'performance portability' and 'data-centric workflows'. Many elements of these foci already exist from developments in the first phase, but their full implementation throughout the entire forecasting system will still require a significant commitment.

[84] National Oceanic and Atmospheric Administration Earth System Research Laboratory
[85] National Center for Atmospheric Research

4 The second phase: 2020-2025

This section will describe the main developments to be undertaken in the second phase of the Scalability Programme between 2019 and 2024. While not all developments are constrained by the next HPC procurement, the timeframe 2024-25 represents the end of the present ECMWF strategy and therefore provides a sensible milestone. As already concluded in Section 3, the main implementation strands will focus on data-centric workflows and on performance and portability of data assimilation and model.

4.1 Data-centric workflows

The data collected, produced, disseminated and archived by ECMWF is becoming larger and denser, and the applications which attempt to handle this data tend do so in an idiosyncratic, segregated manner. In phase one (Section 2.1) many of these applications were improved with regards to data handling, but they were improved in isolation. There are much bigger gains to be made if we recognise that our workflows at ECMWF should be optimised based on a harmonised data model, and not just on the individual applications. This is the concept of data-centric workflows, which take the view that applications should be replaceable and interchangeable, because everything subscribes to the same data model, and that these individual applications should be decoupled from the responsibility of managing data. This brings advantages in terms of performance and flexibility, but also brings a valuable separation of concerns - allowing applications, such as the IFS or PGen, to focus on their main responsibility, and not on data handling.

One of the core components of the data-centric vision is a coordinated, centralised data storage service, which is flexible, performant, extensible and configurable. Version 5 of the FDB, described in Section 2.1.4 meets these requirements by providing an efficient means of storing and indexing curated meteorological data. It is therefore the foundation for the construction of data-centric workflows at ECMWF. The FDB also introduces the concept of data routing, which is central to our strategy to optimise end-to-end workflows. Data routing in the FDB allows selection of the most efficient storage mechanism for different types of data, according to a given configuration. For example, operational forecast data coming from IFS may be directed to use the fastest hardware, whilst research experiments are directed elsewhere. The flexibility and configurability of the data model allows all of this to be done without modifying a single line of application code.

Fig. 36 shows the vision for data-centric workflows, which introduces new developments such as a centralised store for observations (ODB Store); further improvements to the IFS I/O server and MultIO; capability to ingest observations from alternative pipelines (e.g. IoT sources); and the capability to add new data pipelines for alternative data handling (e.g. using NVRAM or delivery to the Cloud). It also includes key developments from the first phase (e.g. SAPP, FDB, see Section 2.1).

4.1.1 Observation handling

Observations form the core input of an NWP system, and the observation handling and assimilation systems form a significant portion of the competitiveness of ECMWF's forecasts. Both the volume of observations that can be processed and the complexity of the observation handling and assimilation pipeline will increase dramatically.

Independent of the native data format, observations are converted into the odb2-format prior to ingestion into the system but also for late storage in MARS. The software tool for handling, encoding and decod-

ing these observations, called ODC, is being developed for further dissemination through 2019. This software provides a significant performance boost over previously released odb2-tools, and comes with a straightforward API that can be used from C, C++ and Fortran.

As part of the strategy to enable external entities to provide observations in odb2-form directly, and to simplify the use of odb2-data for new users, a Python interface to ODC will be released. This will come with a pure python module with the same interface such that data can be provided and used without having to build and install compiled software.

Crucial for increasing the flexibility of odb-type data handling is the introduction of a centralised service for storing and indexing observations for maximising the flexibility and resilience of the data pipeline. This store will operate in the same manner, and use the same underlying technology, as the FDB does, for storing output meteorological fields.

This observation object store (or ODB store) will allow for a more flexible set of workflow components to provide, process and manage observations. Observations will be inserted into the store from a range of sources. Analysis tasks can read data from this store on-demand, but also write observation feedback and departures back to the ODB store for further use down the pipeline and prior to archiving. This architecture will allow transparent changes to be made to how the data is stored, collocated and transported around the system - all behind a stable API. This includes the addition of additional filtering and pre-processing tasks, as well as developing more continuous pre-processing and assimilation systems.

At present, observations are extracted and pre-processed by SAPP. SAPP filters, converts and standardises the data prior to feeding the forecast pipeline. This will continue and data will be fed directly into the ODB store. The flexibility provided by the ODB store will be particularly relevant for ingesting and preparing data originating from IoT sources, as these cover a wide range of novel observations recorded by devices such as mobile phones, cars sensors and personal weather stations. These observations form a firehose of incoming data, which is diverse, unstructured and of widely varying quality and reliability. The scientific software to handle this data is yet to be developed, but for it to work a reliable and scalable infrastructure for handling unstructured observations will be required. The ODB store will play a central part in this infrastructure - allowing scalable access to labelled, unstructured observational data in odb2-format originating from and further advanced for a flexible range of producer and consumer tasks.

The use of such data comes with ethical (e.g. concerning the privacy of mobile phone and car sensor information) and public access (e.g. availability through private companies) issues that need to be

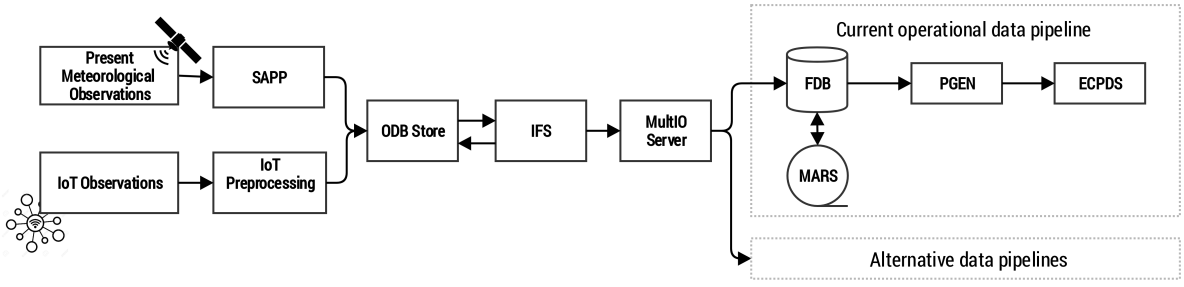


Figure 36: A vision of data-centric workflows at ECMWF, illustrating added flexibility and interoperability between core components. It includes the similarly designed ODB and FDB centralised datastores; and important developments to data routing software - including the MultIO server, which will operate as an I/O server and data routing service for IFS-ST, IFS-FVM, NEMO, WAM & COMPO.

eventually addressed by the meteorological community. This issue is beyond the scope of this paper.

4.1.2 Model output and data routing

Meteorological fields in the NWP-system are distributed geographically, with each geographical region representing a three-dimensional block of the Earth system. Presently, these fields are archived and stored as two-dimensional spherical slices of the atmosphere. This transformation, as well as routing the data from the model to one of the available storage technologies, is the responsibility of an I/O-server.

The current operational IFS has its own I/O server (see Section 2.1.3). It supports the routing of atmospheric fields and has been recently adapted to support fields from the wave model. It is, however, closely coupled to the IFS-ST and other models being developed at the Centre, so that IFS-FVM and COMPO, which offer using different grids cannot easily make use of it. Also, the NEMO ocean model only has NEMO-community support for a different I/O server (XIOS[86]), which is heavily reliant on the NetCDF data format and thus would be difficult to adapt to outputting GRIB. The experimental software

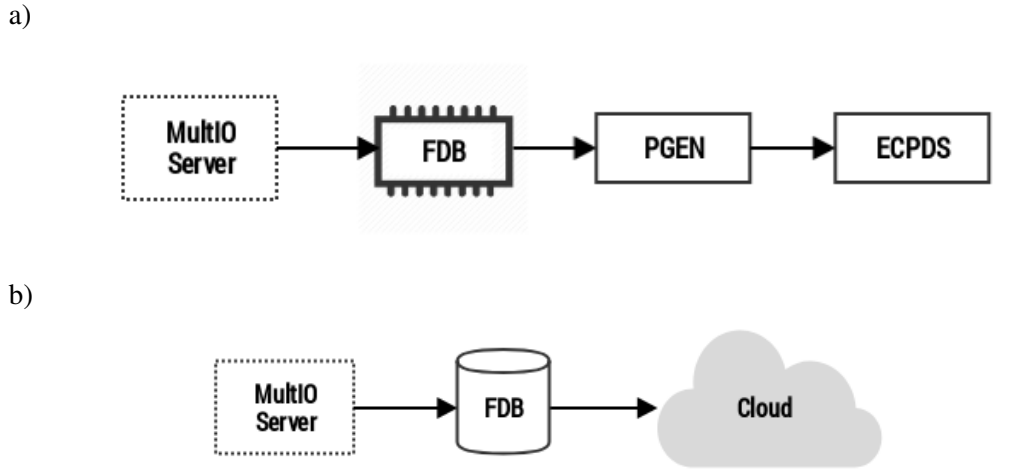


Figure 37: Generalised MultIO I/O-server feeding, for example, (a) model output data for PGen using the FDB executed on NVRAM memory extensions (b) delivering data to users on ECMWF's Cloud infrastructure.

component that will be able to funnel output data to the available storage technologies (e.g. product generation on NVRAM, serving data to the Cloud, etc., see Fig. 37) is called MultIO. MultIO will cover the responsibilities of the different I/O servers in operations now. It will be designed to be general-purpose in the context of Earth-system modelling and to support output from current and future model components including the NEMO ocean model, IFS-FVM and COMPO. This new MultIO functionality is being developed as part of the MAESTRO project (see Section 7.1), which explores new avenues of data movement across multiple levels of the memory and storage hardware as well as the software stack.

In particular, the MultIO I/O-server will be designed to include the following functionalities and technologies:

- a transport layer: to decouple the technology used in data routing and model output from that of used by the NWP system;

[86] XML-IO-Server

- data-format encoding: to support encoding in different formats (GRIB, NetCDF, etc.) and decouple encoding from data modelling;
- a data sink: to make it easily configurable to what storage technology the data is sent to, including middleware abstractions that take over the responsibility of managing different hardware resources.

4.1.3 Convergence of HPC and Cloud services

ECMWF currently relies on two data delivery models. Through ECPDS data is 'pushed' from the data centre to the users. Via MARS, users can also 'pull' data from our archives. As size and diversity of the data increase the processes of moving primary data becomes a limitation, possibly forcing users to process less data than they require or accept delays.

Beyond the plain pushing and pulling of data, ECMWF is pursuing a third model - the Cloud. In this approach the user applications will be brought closer to the data and moving large volumes of primary data away from the data centre is avoided. By providing users with computing resources that are collocated with ECMWF's data sources, accessibility to the data can be substantially improved. This will provide greater value for the existing users (internally and externally), attract new users, and generally increase the scientific capabilities of the Centre.

This view is aligned with the current approach championed in the HPDA [87] community of data-centric computing where the mantra is 'move the compute, not the data'. This principle is applied already for selected Member-State applications today, but there is no generic, flexible framework to support the integration of these applications.

Moving the compute is the core principle of the Cloud: instead of delivering a product, such as a computer or a piece of software, the user can visit the Cloud and use those products in-situ, as services. ECMWF is creating a Cloud ecosystem, the so-called European Weather Cloud, which will provide infrastructure- and platform-as-a-service (IaaS and PaaS), allowing the collocation of HPC and DHS [88] (the data sources) with Cloud computing (the data sink), as shown in Fig. 38. However, this proximity does not solve the data provision issue entirely. To make the crucial link between the data source and data sink, ECMWF's Cloud infrastructure must be connected to a data I/O pipeline, offering a Data-as-a-Service (DaaS) system to ECMWF's Cloud users.

Across many industries, the rise of Cloud computing is disrupting HPC-based ecosystems - both technologically and culturally. The Cloud industry dominates scientific developments in artificial intelligence, IoT and Big Data, and provides an excellent platform for merging data sources and performing dynamic analysis. On the other hand, centralised HPC ecosystems lead in terms of scalable numerical computations, but are less flexible and too costly for smaller institutes to acquire and maintain. There is a strong effort towards a convergence of HPC and Cloud services across these industries, which focuses on bringing these two ecosystems together, combining their advantages and alleviating respective disadvantages. One key aspect of this convergence is the facilitation of seamless data movement between HPC and the Cloud. ECMWF's vision for a DaaS system will deliver this complementarity in our domain.

DaaS vision The DaaS system will provide the link between the HPC workflow and the Cloud. It will be akin to a fast cache of our most recent or most important data, and be fed directly from IFS via the MultIO Server, in parallel with PGen and MARS archival, as shown in Fig. 36 and Fig. 39. It

[87] High Performance Data Analytics
[88] Data Handling System

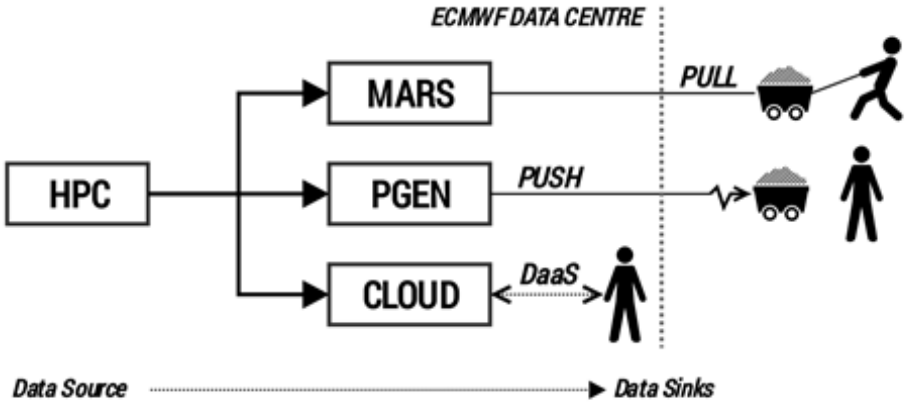


Figure 38: Data delivery from the data source (HPC) to the data sinks (users of MARS, PGEN via ECPDS, and ECMWF's Cloud).

will provide a service which allows access to ECMWF data using Cloud-friendly web standards, and to provide geo-spatial tools for accessing, combining and manipulating data in any direction.

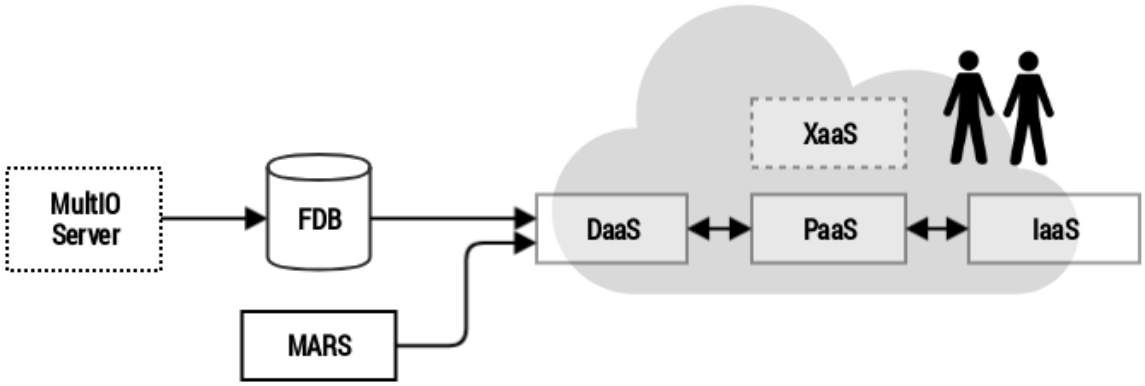


Figure 39: The Cloud pipeline, showing the Data-as-a-Service system as the link between HPC and the Cloud.

The DaaS system must meet certain criteria, namely:

- offer fast access to our real-time data and respect our dissemination schedule;
- support dynamic data access, with no need to pre-register data requirements;
- be scalable, both with size and breadth of data, with the potential to cache operational data;
- not impact operations, in particular it must not throttle the model's progress;
- enable scalable transversal (cross-axis/sliced) data access;
- provide tools for querying and notification of data availability, for scheduling of workflows.

The DaaS system will diversify the usage of ECMWF data and allow the curation of new third-party applications. For example, it will allow a user to efficiently work with cross-axis data, (e.g. time series

data or multi-ensemble data) without transferring large datasets; access data for dynamic situations, for crisis response, emergent events or other inherently dynamic use-cases; seamlessly integrate various sources of Cloud-ready data (e.g. ECMWF's data and Google Earth Engine); and provide on-demand, elastic services to their end-users (e.g. hosting a website or web service of their own).

DaaS implementation The front-end of the DaaS system will be a RESTful API service, to which users can make simple HTTP requests. The main asset of RESTful services is portability between different computing infrastructures on the internet enabled by standardised operations applied to resources available through the Web. With such a service, there would be no need for ECMWF to develop custom clients, because REST clients are flexible and ubiquitous. However, ECMWF could provide client wrappers for major languages (e.g. Python). REST is now a Cloud-standard protocol and it should be simple for our community to use.

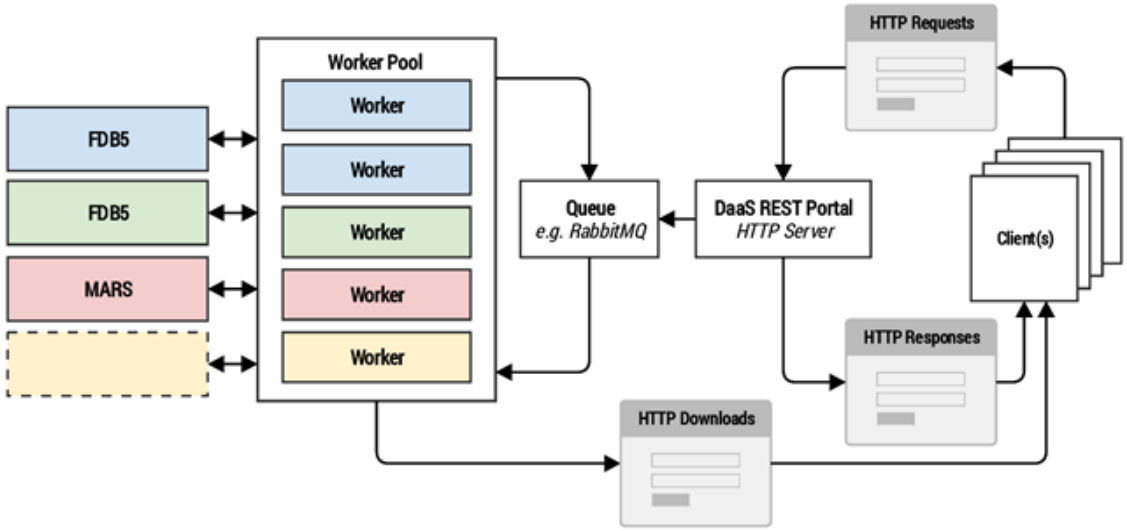


Figure 40: Early concept of ECMWF's Data-as-a-Service system.

The back-end will translate the HTTP requests into data retrieval tasks, mostly targeted at an FDB which stores IFS output in grid-point form and is optimised for geo-spatial data as discussed below (Section 4.1.4). A concept is shown in Fig. 40, in which requests are queued and picked up by worker processes. These processes will interact with the data stores, fetch, transform the data and then serve it to the client. ECMWF will investigate strategies for data distribution and the physical location of the various components. In addition, the best mechanisms to expose data transformation and geo-spatial sub-setting services will be investigated.

Under the Scalability Programme, the revised Hermes project (see Section 7.1) will be responsible for the development of the DaaS system which will serve the European Weather Cloud. Hermes will also act as an umbrella project by ingesting the developments from two EU projects: LEXIS (January 2019 - June 2021) focusing on big data, HPC and Cloud convergence, and HiDALGO[89] (December 2018 - November 2021) focusing on Cloud-based workflows. There will also be a clear benefit for Copernicus.

[89] HPC and Big Data Technologies for Global Systems

4.1.4 Weather and climate data hypercubes

By volume, meteorological field output from the NWP model constitute the largest and most performance-critical part of the data workflow. ECMWF has used the FDB as a software library and operational service to provide the link between the NWP model and product generation, as a location for data prior to archiving in MARS, and as a hot-cache for serving data to meet user requests. As the resolution and variety of data generated from the model increases, and the customer requirements steadily increase as well, an increasing stress is placed on the FDB.

In 2018 the fifth version of the FDB was released, and has been integrated into operational use in 2019 (cycle 46r1). The demands placed on this system are increasing, not only in terms of quantity, but in terms of diversity of data usage. As mentioned above, the FDB technology will support the convergence of HPC and Cloud by acting as the back-end for the DaaS system, and functionality will be added to support the integration of Cloud-facing services. In particular, functionality will be added to deal efficiently with cross-axis data access.

Meteorological objects are currently comprised of two-dimensional slices of the atmosphere, stacked like layers of an onion to describe the atmosphere in three dimensions. They are then described by a number of scientifically meaningful parameters. These include the level in the atmosphere, the valid time and step for the forecast, and which parameter is described. This forms a (conceptually) dense hypercube of data. Although the traditional customer products are largely post-processed from these two-dimensional slices, we aim to support novel Cloud workflows which access the data on-demand across arbitrary cross-sections of the hypercube. This may involve looking at time series data for one point, all the parameters in a vertical column of the atmosphere, or any other combination desired by researchers or users. It is planned to extend the FDB to store and provide direct access to the dense hypercube of meteorological data.

The varied access patterns pose different stresses on the underlying hardware. Novel storage-class memories will provide byte-addressibility of the persistent storage layer. The FDB will be extended to make use of byte-addressable hardware to efficiently slice multi-dimensional data across arbitrary axes. Work carried out in the NextGenIO project has already implemented a first back-end for the FDB which is capable of using Intel Optane DC Persistent Memory, and testing on the prototype system demonstrated an enormous improvement in performance (Section 2.1.4). The back-ends of the FDB will be extended to make use of novel technologies as they become available, and to support more intense and more varied workloads.

4.1.5 Notification system

The IFS currently notifies certain systems when a step of the model has completed, and its data has been flushed to the operational FDB. This is used by PGen, for instance, where the scheduling of complex workflows is done on a step-by-step basis and responds immediately to the availability of data. The Cloud increases the possibilities for scheduled workflows, and many users will wish to create real-time services which run automatically based on the availability of new data. A more robust notification system is required to support the Cloud, which will allow many hundreds of diverse clients to listen and respond to data availability updates.

Since there are now many multiplexed data pipelines, potentially receiving different data from IFS or in different forms, it makes sense to build a more robust notification system which is able to inform relevant downstream services of data availability. It may be possible to delegate the responsibility of notifications

to the FDB, and allow relevant systems (e.g. PGen, Cloud users via DaaS, early delivery to Member States) to receive notifications from the FDB, or it may be beneficial to implement a dedicated service for notification tracking and handling. Decoupling the responsibility for notifications from the IFS will allow greater flexibility, and is closely aligned with the principles of data-centric computing.

The mechanism of notification is to be investigated though. One option would be a typical publish-subscribe service, possibility implemented using RabbitMQ or Apache Kafka. Alternatively, a simpler method using distributed semaphores may be preferred. The notification system will be developed as part of Hermes and the DaaS system, as it is mostly aimed at improving the usability of ECMWF's Cloud.

4.1.6 Workflow performance optimisation and simulation

The *Kronos* workload simulator (and associated set of tools) has been developed at ECMWF as part of the NextGenIO project (see Section 2.1.5). This has involved developing tooling to process, standardise and analyse large-scale performance data from NWP workflows. Model workloads have been generated from this analysis, and these can be used together with synthetic applications and an execution server to execute a representative workload on any HPC machine in a portable fashion.

Both the analysis of the workload data and the generation of synthetic workloads has been labour intensive and relatively manual. It is intended to build tools to automate data collection from suites running on the HPC systems. More significantly, to build a more extensive data modelling machinery, to perform analytics and build meaningful, scalable model workflows from the gathered data. This will allow to gain deeper insight into the performance, bottlenecks and the variability of our forecast systems.

Automation of analytics and analysis will facilitate ongoing monitoring of the performance of the operational workflow. This is important for the early detection of performance issues, and the isolation of their causes. Automated generation of synthetic workloads will help ECMWF to test representative workloads in a range of configurations on novel hardware, and to test modified software components in a meaningful environment. This will continue to be an essential asset for future HPC procurements as well.

4.2 Performance and portability of data assimilation and model

The growing diversity of HPC hardware is making it ever more challenging to achieve high computational performance across a spectrum of architectures, compilers and hardware vendors, while offering new opportunities for increasingly bespoke hardware solutions based on accelerators and novel programming paradigms. Ensuring the future efficiency and scalability of the IFS forecasting suite will require changes to both the algorithmic options available in the software, and to the software implementation of the suites themselves to fully leverage emerging hardware trends without sacrificing scientific performance. Historical algorithmic choices, such as the spectral-transform model with semi-implicit semi-Lagrangian time stepping, should be revisited in light of the changing HPC landscape. New algorithms, better suited to the increasing amounts of multi-level parallelism in modern supercomputers, in the field of numerical methods, spatial discretisation, time stepping and data assimilation, will be investigated (see Section 4.2.2).

Performance optimisation is becoming increasingly complex and bespoke to specific hardware architectures, accelerators and compiler stacks. It is therefore of great importance to decouple the performance-relevant implementation details of the code from the definition of the scientific computation, a software engineering principle known as 'separation of concerns' (see Fig. 41). It is envisaged to prepare the

IFS for emerging HPC architectures through adaptation towards a more modular software infrastructure, where parallelisation strategies, data layout and other performance-relevant aspects are delegated to a software middle-layer that can adjust to the relevant type of HPC system (see Section 4.2.3). Achieving this separation is a crucially important step towards performance portability and will enable early evaluation and a quick transition to existing and novel HPC architectures as they emerge.

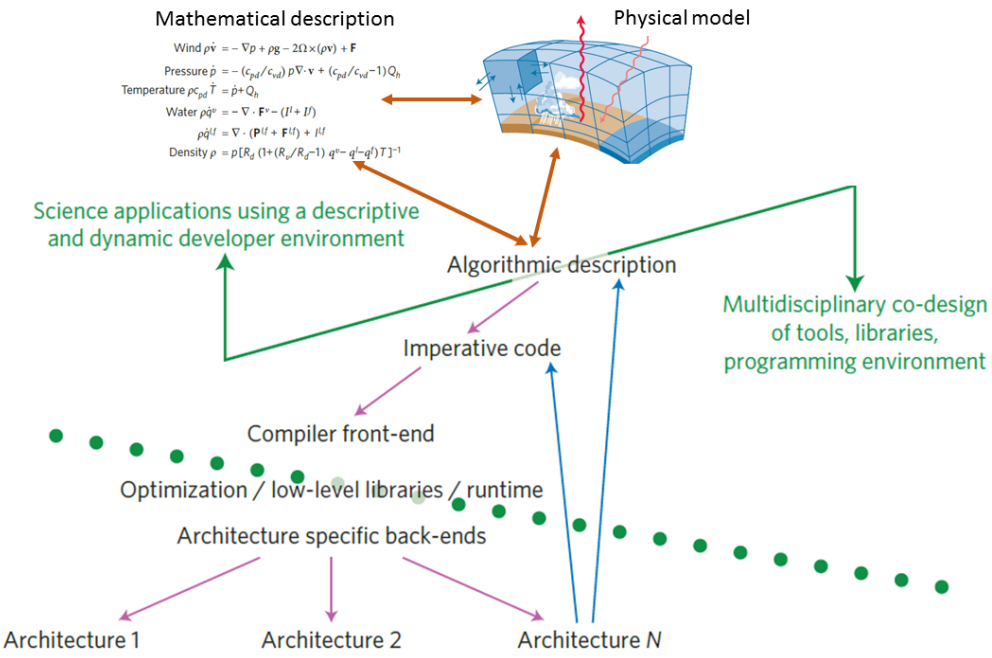


Figure 41: The 'separation of concerns' is the separation of the science-code levels from the hardware-sensitive code levels. The former encapsulates the mathematical description of a physical model (orange arrows), here the Earth system, while the latter translates the available algorithmic concepts to flexibly map memory access patterns, parallelisation and vectorisation onto several hardware back-ends. The two levels are not independent from one another as specific algorithms and numerical methods will have advantages and disadvantages in this mapping process. The separation of concerns requires a tool-chain that translates between these levels, as indicated by the purple arrows (see Fig. 46). However, architectures also drive the choice of algorithms and numerical methods (blue arrows) in performance optimisation (adapted from [95]).

Entirely hardware agnostic implementations of the whole IFS are likely to be very difficult to achieve but a step-by-step approach is planned, where a combination of libraries, source-to-source translation tools and DSLs are used to enable quick adjustments to core implementation choices, such as data layout and loop ordering, throughout large parts of the code. Importantly, this adaptation must include a generalisation of current CPU-specific optimisations, while allowing alternative approaches to be deployed on HPC platforms with different performance characteristics. The development of a comprehensive software infrastructure tool-chain that is capable of targeting and optimising for CPU-based systems, as well as GPU accelerators and new dataflow architectures, is now being pursued in collaboration with ETH, CSCS and MeteoSwiss in Switzerland.

Equally important will be the investment in a much more comprehensive code performance analysis environment that allows to (i) trace performance bottlenecks - amongst others related to data communication, memory hierarchy level access, shared and distributed memory parallelism, load balancing - back to their

roots in both workflows and code; (ii) estimate how efficiency gains will propagate through the entire system. This effort will assume high priority at the beginning of the second phase of the programme.

4.2.1 Performance analysis

As ECMWF readies the IFS code for architectures other than the incumbent CPU, it is becoming increasingly important to have a well-informed and detailed view of performance achieved by the current code, as well as its intrinsic performance-related code characteristics.

With modern HPC tools giving access to hardware performance counters (e.g. PAPI[90] counters for CPUs) at a configurable granularity, it has become possible to establish roofline plots [110] for each of the main IFS components. This involves establishing actual bytes transferred as well as FLOPs, allowing effective arithmetic or operational intensity to be determined for each component. Algorithmic intensity estimation, based on assessment of the code itself rather than its execution, should also be undertaken, as this can provide an upper bound against which achieved performance can be meaningfully compared.

With a comparison of achieved performance to expected performance of the main IFS components, targeted optimisation of the current code base can then be attempted. The analysis of hardware counter information should again guide such optimisation efforts. Derived metrics such as memory usage efficiency, memory boundedness, instruction execution efficiency, load imbalance and more, are all useful indications for directing optimisation efforts.

4.2.2 Exploring algorithms

Following the principle of separation of concerns in software design, the introduction of OOPS has allowed to decouple variational data assimilation algorithms from their model specific implementations. As discussed in Section 2.2, OOPS building blocks can be combined together into applications, in particular the incremental 4D-Var methodology employed in operations. The 4D-Var algorithm in the strong constraint formulation has a purely sequential nature. The parallelisation of each building block is achieved only in the space dimension. In order to ensure performance portability of the ECMWF assimilation software, substantial efforts will be required (i) to ensure that individual building blocks can scale on future massively parallel hardware and (ii) by exploring alternative algorithms/formulations intrinsically exposing more parallelism. While the first task is likely to benefit from the work done on the forecast model itself, the second task is another scientific endeavour. Flexibility of the OOPS-based system will allow to more quickly implement and test new algorithmic solutions in the future. It should be noted that achieving task (i) alone may not by itself guarantee running 4D-Var efficiently and in a timely manner on future architectures in the longer term.

Continuous data assimilation: The more immediate concerns regarding the scalability of 4D-Var can be addressed by exploiting ideas brought forward by the continuous data assimilation framework introduced with cycle 46r1 [69]. The key algorithmic insights here are that a) there is nothing in the incremental 4D-Var algorithm that prevents us from using a different number of observations in different outer loops, and b) that as long as the background state and the background error covariance (B) matrix do not change, one can re-initialise the minimisation from a previous analysis without losing statistical consistency. These two ideas allow for an effective decoupling of the time we start the 4D-Var computation to the cut-off time of the observations used in the analysis.

[90] Performance Application Programming Interface

In this implementation of the continuous data assimilation the number of outer loops has been increased from three to four in both the long window and early delivery analyses, yet it is still possible to meet the operational time constraints because the first outer loop is triggered earlier, without any loss of observations (in fact, overall 5-7% more observations are being assimilated than in the standard 4D-Var configuration). Fig. 42 shows the schematic of the present and new early-delivery configuration. While the window length is fixed to 8 hours, each outer loop sees an increasingly larger number of observations as they become available (observations arrive with a delay due to latency in receiving the data). The

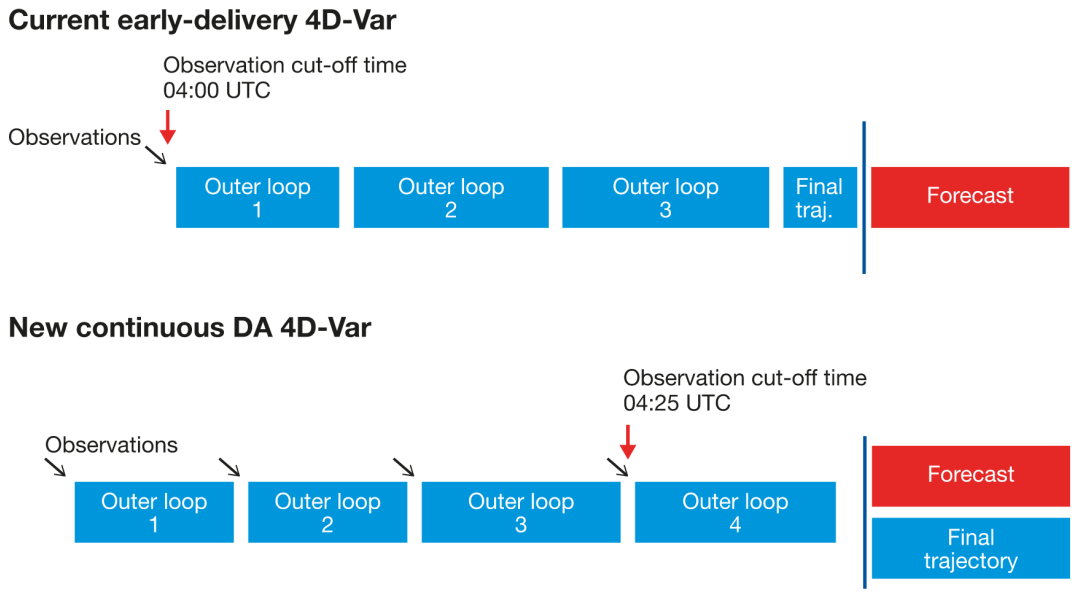


Figure 42: Schematic representation of observation data flow in current (top) and new (bottom) early-delivery 4D-Var configurations, the latter based on the continuous data assimilation approach.

continuous data assimilation ideas can be pushed further, following well-established ideas in the literature [61] [87]. In the current operational configuration of the analysis suite [52], the analysis from the early-delivery, short-window analysis is used to initialise the 10-day HRES forecast and to centre the ENS forecast, but its information is not cycled. The first step towards a continuous long-window data assimilation system is to use the early-delivery analysis to warm start the long-window 4D-Var. In this way, information about the Hessian of the cost function and a more accurate linearisation trajectory obtained by ingesting observations falling in the first 8 hours of the assimilation window can be exploited in the solution of the long-window 12-hour 4D-Var problem. This idea will be first realised with IFS cycle 47r1, and can be further extended. Fig. 43 shows a possible configuration where data assimilation is run as a semi-continuous service. Each consecutive 4D-Var task has an assimilation window extended forward in time in order to accommodate newly available observations, while the beginning of the window is fixed, thus guaranteeing the statistical consistency of the scheme. As each analysis has seen a large fraction of the observations used in the successive analysis, the final trajectory of the previous analysis will be an accurate starting point for the successive minimisation, which is also beneficial for the exploitation of non-linear observations. Additionally, each successive 4D-Var task would require fewer outer loops to converge to solution.

Such a system would have the additional advantage of distributing the computational work load more evenly in time. In this set-up, only the last outer loop would be in the critical path and one could attempt to accommodate the start of the computation to fit other operational requirements. Concerns about the time-to-solution of the entire incremental 4D-Var algorithm would thus be reduced, to a large extent, to

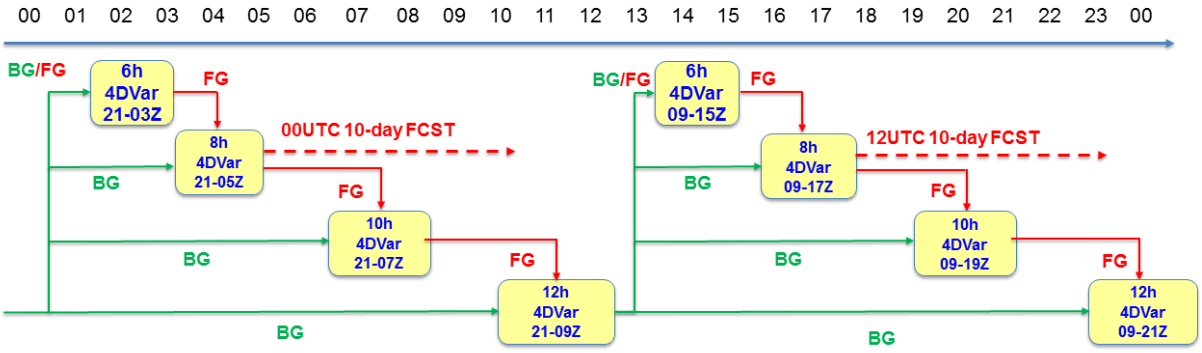


Figure 43: Schematic representation of an implementation option for a continuous long-window data assimilation (BG = background, FG = first guess, FCST = forecast).

concerns about the time-to-solution of the final minimisation. While this is not an immediate concern, it can be expected that increases in inner loop resolution, and consequent time-step reductions, will at some point require to investigate minimisation algorithms that are parallel at the inner-loop level.

4D-Var components: Hardware portability of data assimilation has received little attention until today, as in recent years the resources were primarily allocated to development of the OOPS system. In variational methods the model, its tangent linear and adjoint versions are complemented by other operators including the background error covariance, observation error covariance, non-linear observation operator and its linearised counterpart. Ensuring hardware portability of data assimilation as compared to the model can be therefore seen as a more complex task, as it needs to account for all the other components.

Fortunately the profile of 4D-Var, as illustrated in (Section 2.2) is very steep with the tangent linear and adjoint model representing the bulk of the cost and the remaining operators are relatively insignificant. One could therefore hope that leveraging the developments done for the non-linear model in its linear counterpart could be sufficient. Unfortunately, if only the linear model is adapted to alternative architectures or accelerators, expensive data movements in model space will be required before applying other operators. Ensuring the hardware portability of the B-matrix while keeping the operators that act in observation space initially intact, would allow to limit the required communications to observations space, which is of much lower dimension than the model space, as only the model values interpolated to observation locations would need to be communicated. Such approach, while benefiting from the non-linear model work in adapting the linear and adjoint models and addressing a relatively concise implementation of the B-matrix, will avoid complex adaption of the observation space operators, which can be addressed at a later stage when resources become available. It should be noted that the observation operator relies on a 3rd-party radiative transfer model (RTTOV [91]) making its adaptation more challenging.

Eventually all the components of the control variable, including variational bias correction, skin temperature sink variable and weak constraint will need to be adapted to run on future hardware. An important requirement for data assimilation software adaptation is preserving the symmetric positive definite property of the Hessian of the quadratic cost function, as it allows to employ an optimal minimisation algorithm that guarantees global convergence for the convex optimisation problem at hand. In practice it means that the products of tangent-linear and adjoint operators are symmetric to the machine precision. This is achieved by writing the adjoint code to reflect exactly the sequence of operations in the tangent

[91] Radiative Transfer for TOVS

linear but in reverse order. The design of a DSL should cater for that requirement.

4D-Var algorithm: Addressing the algorithmic scalability bottleneck of 4D-Var is far from trivial. OOPS has enabled us to start evaluating the ideas put forward in the recent ECMWF Data Assimilation Strategy [13]. The saddle-point formulation of the weak-constraint 4D-Var [41] entails poor and non-monotonic convergence issues [47][48]. The sliding-window cycling for weak-constraint 4D-Var, which can be approximated by overlapping assimilation windows that move forward in time by one sub-window at a time [42] is now considered to be challenging to implement in practice for a high dimensional model like IFS, as it requires the availability of the inverse of the square-root of the background error and model error covariances. Such operator exists in the IFS. An attempt has been made to test a simpler config-

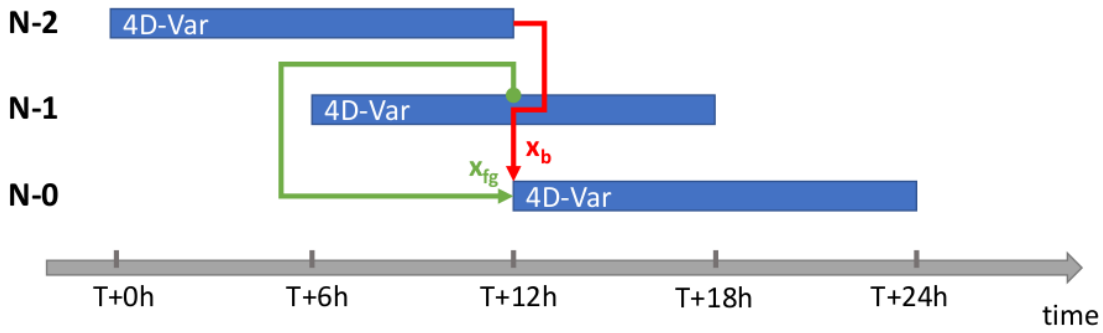


Figure 44: Overlapping 4D-Var configuration. The N-0 cycle is initialised using the background (x_b) obtained from the N-2 cycle and the first guess (x_{fg}) is obtained from the N-1 cycle, respectively.

uration first: an overlapping strong-constraint configuration is shown in Fig. 44. In this configuration the 12-hour 4D-Var overlaps over the first 6 hours of the previous cycle. In order to preserve statistical consistency the background state for the N-0 cycle is taken from the end of the window of the N-2 cycle. One can expect that a first guess obtained as an analysis from the middle of the window of the N-1 cycle, which has seen observations between 6 hours before and after the beginning of the N-0 cycle's window, would be closer to the N-0 cycle's analysis and thus allow to reduce the required number of outer and inner iterations.

In order to compute the first-guess gradient of the quadratic cost-function correctly, it is required to apply to the inverse of the adjoint of the square root of the background error covariance an increment computed as a difference between the first guess and the background. Unfortunately this leads to divergence of the Lanczos algorithm [46]. There are two possible explanations. Firstly, the first guess is a bad starting point and, secondly, it has been found that the spectrum of the first-guess increments is very different compared to a typical spectrum of increments obtained through minimisation, which may render it incompatible for the minimisation. The latter explanation is plausible, as the case where the first guess increments are obtained through the first minimisation and the first-guess gradient is computed by applying the inverse of the square root of the background-error covariance works and gives almost identical results as the case where one simply carries on with the second minimisation.

Further effort will be required to understand this behaviour before considering the sliding-window weak-constraint formulation. Nevertheless, testing the latter for a simplified model could still be of interest to understand further its strengths and limitations. Research on the second-level preconditioning, as described in the strategy, is a promising path. Preconditioning vectors to be applied in the first minimisation can be precomputed outside of the critical path by exploiting the fact that the Hessian depends on the trajectory, but in most cases does not depend on the values of the observations, just their locations. The

latter can be predicted with a large degree of accuracy.

More advanced, second-level preconditioning algorithms proposed by [101] and tested for simple models, will be implemented and evaluated. OOPS will enable us also to explore a new promising strand of research devoted to techniques to approximate the solution of a high-dimensional Bayesian problem based on optimal low-rank projections that maximise information content of the Bayesian inversion. Such optimal projections can be constructed by using embarrassingly parallel stochastic SVD algorithms [12][15]. It is not known at this stage how many singular vectors are needed to characterise the information content of observations for a high-dimensional and densely observed system like the IFS and what would be the total cost to solution of such methods. Previous experience with similar ideas, like the Reduced Rank Kalman Filter [40] and the Ensemble-Variational Integrated Localised (EVIL) Data Assimilation [7] suggest that a purely stochastic but parallel approach might be too computationally demanding to be of practical use. At this stage of research, we believe that a hybrid approach combining stochastic SVD with gradient information is the most promising research direction. This will be pursued first in the simplified-model-test environment available in OOPS.

The design of the OOPS system may need to change further in the future depending on the scientific requirements and hardware constraints. The current OOPS design aims at eliminating I/0 and initialisation costs of the 4D-Var algorithm by running outer and inner loops from a single executable. Such approach brings tangible efficiency gains already today. This, however, may not be necessary in the future. One of the reasons is that with the upcoming NVRAM technology (see Section 2.1 and Section 4.1, the I/O cost is likely to become insignificant.

One may also consider advantageous an ability to optimise the computational configuration of the outer and inner loops separately as they are set to have different performance sweet spots (since they are executed at different resolutions). While this may not matter in operations where the unused resources would likely remain idle, these optimisations may result in increased throughput for research. Today, the total run-time of the inner loops at operational resolution or in research experiments is much longer than the total run-time of the outer loops. Future configurations may see an increased gap between the outer- and inner-loop resolutions due to the scalability bottleneck of the inner loop. This will bring the run-time of the outer loops much closer to the run-time of the inner loops. At this point one may want to optimise computational layout for both separately. It is also expected that in the continuous data assimilation system described above, fewer outer loops would be required than in today's operational configuration. This means that initialisation and I/O cost would represent a relatively small fraction of the total 4D-Var cost.

Coupled data assimilation: Similar concerns regarding the present OOPS-design also arise for the coupled data assimilation system. With the current OOPS-design, it would be relatively straightforward to implement a strongly coupled assimilation system by developing a coupled implementations of the OOPS building blocks. In this scenario the control layer would not need to be adapted. However, the intermediate step on the path towards full coupling, the so called outer-loop coupling [68] would not fit in the current OOPS framework today. OOPS would need to be able to evaluate a coupled non-linear cost function and set up two independent quadratic cost functions for the atmosphere and for the ocean that would be minimised separately. This means that OOPS would always need to be aware of the two Earth-system components. The two components would also need to share resources including the hybrid parallel layout unless new technologies facilitate adapting the computational layout at run-time for sub-sets of parallel tasks.

Resiliency questions also arise. In particular, the question is how to deal with failures in one component

of the coupled system affecting the other components in an operational setting. The relationship of the coupled model and the OOPS control layer is not yet well defined. Currently it is the coupled model that controls the time stepping and the flux exchanges of its components. On the other hand, OOPS controls the time stepping of the model. Decisions will need to be made where the control of the time-stepping and coupling should be. While there are many questions regarding the design aspects of the coupled system today, the OOPS control-layer is considered flexible enough to accommodate all required solutions. In the short term, OOPS can be adapted to run only a single outer-inner-loop pair per task to facilitate the implementation of the outer-loop coupling. Such approach would postpone the requirement of interfacing NEMO to OOPS and also allow to perform the inner loop for the ocean component in a separate executable. The OOPS control-layer would only need minor modifications for this implementation. However, the price to pay may be reduced performance and reduced flexibility of the system. The latter point applies in particular to what is perhaps a still distant future, i.e. a strongly coupled assimilation system.

Regarding the coupled model components themselves a first step towards leveraging the approach planned for the atmosphere would be to implement the tri-polar ORCA[92] grid in *Atlas*. Once this is achieved the same DSL tool-chain can be used to abstract away hardware specific constructs. The introduction of *Atlas* in the ocean model will also facilitate the exchange of fluxes at the interface of the coupled system and can be considered a key requirement for the development of an efficient, strongly coupled data assimilation system. Independently, ESCAPE-2 already investigates the application of DSL tool-chains to the NEMO model.

Atmospheric model development: As shown in Section 1.2.3, the IFS-ST dynamical core with SISL integration of the hydrostatic primitive equations remains competitive at resolutions finer than 9 km. The octahedral reduced Gaussian grid circumvents the computational expense of the latitude-longitude coordinate framework where the meridians converge towards the poles, while the regular, quasi-uniform distribution of nodes on the surface of the sphere facilitates alternative dynamical core options. However, with finer resolutions the communication cost from global spectral transforms and semi-Lagrangian halo exchanges gradually becomes a significant part of the total computational cost. The alternative development of IFS-FVM and *Atlas* addresses this concern, and has been described already in Section 2.2.2 as it was a major achievement along the first phase of the Scalability Programme and a demonstration of the necessary interplay between applied and computational science developments.

Obviously, there are benefits from considering an approach where both spectral-transform and finite-volume dynamical cores can be run on the same grid, delivering a hybrid, unified and switchable modelling system that facilitates direct comparisons of different dynamical cores, fully-integrated with the same physics. It provides a tool that can give clear answers to scientific and technical questions regarding the most suitable options for the dynamical core given future forecast suite configurations and HPC architectures. Although, there are known benefits for IFS-FVM, e.g., reduced volume of data movement, the spatial resolution that we anticipate to outperform IFS-ST in terms of efficiency and accuracy still remains an open question which can only be answered by direct comparisons across scales. This is in part due to the recent advances in the spectral transform acceleration on hybrid GPU machines such as Summit.

A first example of the parallel efficiency gain with IFS-FVM over IFS-ST dynamical cores is shown in Fig. 45 at 13 km global resolution. This test will be extended to include the physics and at better spatial resolution to assess where the cross-over point lies in terms of time and energy to solution. These runs

[92] NEMO global ocean configurations that are run together with the LIM sea-ice mode

form part of the INCITE proposal for Summit. Performance tracing output using the BSC Extrae tool will also be processed by the POP-2[93] centre of excellence for a deeper analysis to check the reliance on communications (volume of data communicated, latency issues), parallel efficiency, and the number of instructions required per time step.

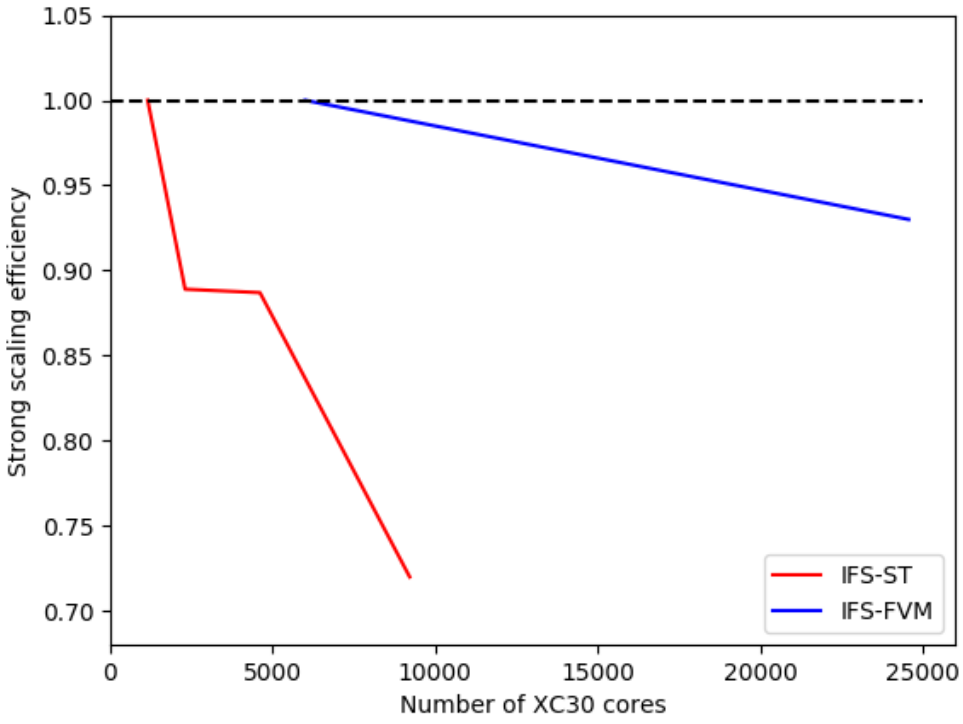


Figure 45: Strong scaling efficiency of IFS-ST and IFS-FVM dynamical cores, following the protocol of [76] for 13 km workloads to which IFS-ST also contributed as a guest model.

As the scientific development of this hybrid system is nearly completed, the bulk of the work in the second phase of the Scalability Programme will focus on the development of the tangent-linear and adjoint versions of IFS-FVM, and the implementation of the separation of concerns for both IFS-ST and IFS-FVM (described in Section 4.2.3).

Local and inherent conservation of tracer advection with IFS-FVM opens new operational and research options for CAMS. The current lack of mass conservation, although mitigated to some extent by the application of global mass fixers, is expected to have a detrimental impact on the realism of multi-year simulations of long-lived greenhouse gases such as CO_2 because the conservation errors can be of similar size as the considered fluxes and emissions. The application of mass conserving flux-form advection scheme might be too costly for operational applications of the multiple tracers of chemistry-aerosol configuration but mass conserving reference simulation could be used to optimise the global mass fixers used for the SL-advection. More details on atmospheric composition can be found below.

Coupled models: The IFS wave model WAM has a long history of code development, but unlike the rest of the IFS code which has been ported to 'modern Fortran' (F2003/F2008 standard), the wave model

[93] Performance Optimisation and Productivity

is still mostly written in 'fixed format' (Fortran 77) and can benefit from a serious code modernisation effort. The timing for this coincides with tackling its scalability issues, and the plans for code-adaptation towards GPU readiness.

As the wave model is currently not used outside of ECMWF, a rather aggressive code modernisation strategy can be pursued without impacting co-developing partners. WAM can essentially be seen as the combination of wave advection and parametrizations. The parametrizations - like the other IFS physical parametrizations - are grid-agnostic and local to one location in space, whereas the advection requires stencil-aware data structures. Currently the wave advection scheme assumes that the grid is a reduced grid, similar to the IFS atmosphere grid, but not the same as latitudes are regularly distributed (i.e. not Gaussian).

The resolution and domain-decomposition are also different than the atmospheric grid to maximise load-balance. This comes however at the cost that coupling (re-gridding) between the wave model and the IFS atmosphere model requires significant parallel communication. To avoid most of this parallel communication at the cost of a certain load imbalance, the wave model could be adapted to instead use either the same grid and domain decomposition as the IFS atmosphere, or a sub-sampled grid sharing the same domain decomposition.

By creating data structures based entirely on *Atlas*, the wave model could become more future proof to changing grids, and use *Atlas* to couple the atmosphere with the wave model. Further parallelisation routines such as halo-exchanges that are currently present and bespoke to WAM could be removed and delegated to *Atlas*. As an added benefit, *Atlas* provides a solid foundation for GPU adaptability and parallel efficiency. With respect to parametrizations, a similar approach to the IFS physical parametrizations is proposed, where it can be written in a domain-specific, single-column abstraction. The wave model parametrizations will then be made performance-portable by a DSL tool-chain with back-ends targeting specific hardware solutions. Our goal is to completely abstract the grid and advection code from the wave model so that other advection schemes can be slotted in easily, which may be either written in a DSL for performance portability or custom tailored for a specific hardware solution.

Atmospheric composition: Reducing substantially the overhead of atmospheric composition simulation as part of medium-range, monthly or seasonal applications would make it possible to simulate weather-composition feedback. Besides general optimisation efforts, two approaches will be pursued: (1) Dual-grid configurations in which the composition is simulated at a reduced horizontal resolution and (2) the introduction of simplified composition schemes. The main challenge for multiple grid simulations will be to maintain dynamical and physical consistency between the representation of processes such as convective transport on the high and low resolution grid. The simplification of schemes can be achieved by only considering individual components such as desert dust or by the substitution of costly chemical solvers with machine learning approaches. While (2) still requires substantial development work, several simplified schemes are already configurable in the IFS.

The simulation of the chemical reactions requires the solution of a numerically stiff system of coupled ordinary differential equations (ODE). First-order (Eulerian backward) and second-order (Rosenbrock) solvers have been implemented in the IFS, which are computationally demanding and require high numerical precision. Therefore, running IFS-COMPO in single precision has not been successful so far, but will be investigated further.

The simulation of the chemical mechanisms can be greatly parallelised as no communication between grid boxes is required. There have been several studies showing the benefit of using GPU accelerators for the simulation of atmospheric chemistry [2] although challenges still remain because of load imbalances.

The usage of the KPP[94] package, which is widely used in the atmospheric chemistry community to machine-generate computer code for chemical mechanisms, was recently introduced in CAMS. It will help to benefit from external developments on chemistry code portability and solver development.

More recently, machine learning techniques have been tested to substitute the numerical solvers of the chemistry schemes. Although there seems to be great potential for performance gains, these have not yet been demonstrated [62]. Using simplified approaches including machine learning is currently considered as one option to introduce the tangent linear and adjoint representation of atmospheric chemistry in the IFS (see also Section 4.3).

For the simulation of atmospheric composition in model configurations (to enable the simulation of weather-composition feedback), the reduction of I/O volumes and frequency as well as the application of reduced aerosol or chemistry scheme are available options to reduce the computational cost of the composition simulations.

4.2.3 New programming models

Preparing and adapting the IFS to increasingly diverse computing architectures often requires very invasive changes to the fundamental structures of the code, such as memory data layout and loop structures, without changing the scientific essence of what is being computed. Since the set of performance optimisations needed to fully utilise emerging architectures are also becoming progressively bespoke to individual classes of accelerators (GPU, FPGA) it is vital to separate the definition of the scientific computation from all performance-relevant implementation details that are likely to change between the current CPUs and any alternative architecture types. Moreover, separating performance engineering tasks from scientific development will allow continuous research efforts into state-of-the-art optimisation techniques and novel programming paradigms without impacting the productivity of domain scientists.

To achieve this separation of concerns between domain scientists and performance engineers, the introduction of a software middle-layer is proposed that provides high-level software abstractions and interfaces to adapt data structures and loop hierarchies underpinning the scientific sub-components of the IFS. Such an infrastructure will allow a more flexible adaptation to alternative execution and memory access patterns in order to fully utilise specialised accelerator architectures (eg. GPUs), while providing equivalent optimisation patterns for cache-based CPUs architectures that match the current performance. Eventually, an exploration and incremental conversion to novel programming paradigms, such as dataflow or spatial programming [77], necessary to utilise dataflow architectures, such as FPGAs (see Section 2.3.2), will also become possible.

Since modern HPC architectures are mainly constrained by the cost of data movement [105], the primary concern of the proposed software middle-layer is the optimisation of data locality and scheduling data accesses across multiple layers of the memory hierarchy. This necessitates that data allocation in memory is delegated to the infrastructure layer, while the necessary parallelisation, data access and loop scheduling choices made in performance-relevant code sections are adapted accordingly. A combination of library interfaces, source-to-source translation tools and DSLs can be used to incrementally introduce high-level programming abstractions, such as stencil-based kernel programming or single-column format, into the individual model components of the IFS that achieve this delegation, as described in the following sections and depicted in Fig. 46.

[94] Kinetic PreProcess

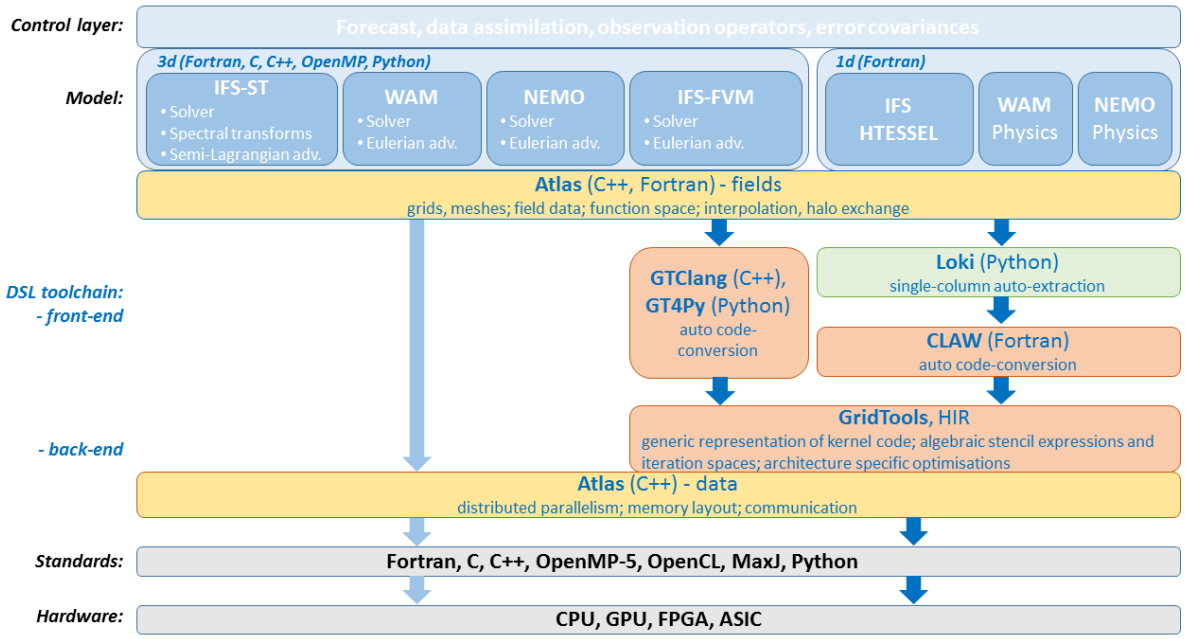


Figure 46: Structure and components necessary for the transition of the IFS to separate applied science from hardware sensitive code levels (see Fig. 41). The light blue arrows indicate the development path for IFS-ST without using the DSL tool-chain, while the dark blue arrows denote the full implementation of the tool-chain for IFS-FVM. For details see text in this section and in Section 5.

Role of Atlas: A key structural change to prepare the IFS for future HPC architectures is to migrate the primary field arrays underpinning the central components of the code to higher-level data structures that act as data containers and are capable of adapting to the various memory hierarchies exposed by emerging HPC architectures. The *Atlas* library (Section 2.2.2) has been chosen to provide this data-management feature as it provides a statically compiled API based around the core concepts of fields and function spaces that allows model fields to be defined without necessarily committing to a specific data layout in memory or defining the placement of data in a complex memory hierarchy.

Using *Atlas* objects to encapsulate field data provides the ability to delegate multiple parallelisation concerns, such as domain decomposition with overlapping halos and parallel communication patterns to a dedicated library. Since *Atlas* objects abstract and encapsulate grid- and domain-decomposition knowledge from the IFS they can also provide parallel communication patterns, such as halo exchanges, as well as gather and scatter operations. *Atlas* manages the memory of field data allocations covering the halo regions in the horizontal, thus acting as a dedicated data management API. As such, *Atlas* ultimately manages data storage alongside appropriate metadata and provides a mechanism to clearly express the data dependencies between multiple components within the code, where the interfaces to relevant sub-components are defined in terms of groups of *Atlas* fields, rather than raw Fortran arrays. This ultimately allows the explicit management of data offload to accelerators, including optional adjustments to data layout, which in turn enables rapid adjustment and performance exploration to dedicated accelerator architectures without compromising current CPU performance.

In order to remain performance neutral on current CPU-based systems, the integration of *Atlas* fields needs to complement the current block-parallel execution model employed in the IFS. For this purpose we plan to use the concept of a 'memory view': a sub-array of the process- or thread-local portion of a

field array that is used during a single iteration of thread-parallel code section, while the underlying data layout of the array is left to be controlled by the container object. Since a large part of scientific kernels in the IFS are already adapted to working on individual thread-local sub-arrays, only a limited number of changes to most scientific routines are expected, restricting the key integration work to the control flow routines that invoke these kernels. As a consequence, the *Atlas* field API needs to provide equivalent array blocks (views) at run-time while providing a memory allocation mechanism that replicates the current block-strided memory layout.

Additionally, the migration of core data structures to *Atlas* objects will allow for a clearer expression of the overall control flow and will provide various grid-related data transformations, such as re-gridding and interpolation or spectral transforms, under a concise API. The use of *Atlas* field objects will also encourage a clearer definition of interfaces between various model components as well as data dependencies between them. It will also provide the opportunity to integrate a data object hierarchy which better maps to the OOPS interface. Moreover, multiple individual implementations explicitly optimised for specific accelerator platforms can be integrated within such an API framework, allowing a more seamless integration of individual porting efforts into the IFS code base.

DSL and tool-chains: In addition to data layout changes, the high degree of code specialisation required to achieve competitive performance on emerging accelerator architectures often entails significant changes to the loop structure of the kernel code, many of which can prove detrimental to CPU performance [22]. In order to overcome the limitations of compiled general-purpose languages like Fortran or C/C++, DSL can be used that encapsulate high-level computational concepts to generate highly-optimised code bespoke to individual HPC architectures. These languages are not restricted by the conservative assumptions on application semantics and data management that traditional compilers have to adhere to and therefore often provide advanced optimisation techniques, such as stencil computations, advanced loop scheduling (polyhedral analysis, loop tiling) and explicit computation-communication overlap to target multiple HPC architectures from a single source code [38, 49, 90, 1, 89, 9].

What is a DSL tool-chain?
A DSL tool-chain is a set of tools that generates platform-specific, optimised source-code from high-level expressions that encapsulate specific computational or scientific patterns not expressible in general-purpose programming languages like Fortran or C/C++.

In order to flexibly adjust a large number of scientific subroutines to alternative execution strategies based on these platform-agnostic concepts it is important to decouple the scientific definition of individual components of the IFS from their low-level implementation. The GridTools framework (formerly STELLA [50]), currently used operationally by MeteoSwiss for GPU execution of the COSMO model, was chosen for this purpose, as it supports multiple abstractions relevant to the development of NWP codes through various front-ends and source-to-source translation tools [49, 45, 22]. In addition to its current GPU capabilities, an ongoing effort to generalise the underlying infrastructure is underway to increase the applicability of the framework and thus enable future extensions to novel emerging architectures [84].

Due to the multitude of computational concepts employed throughout the IFS, several implementation strategies will be explored to convert individual sub-components to best express existing code in the appropriate format. Two general themes have emerged here that fit well with proven concepts within the GridTools infrastructure:

- **Grid-point stencil computations:** three-dimensional solver algorithms in grid-point space can be expressed as a combination of stencil-like computation patterns and halo exchanges, allowing optimisation techniques, such as loop reordering and loop fusion to be applied during architecture-specific code generation for GPUs. Multiple front-end formats will be explored for this purpose that embed the core GridTools API in C++ (GTClang) or Python (GT4Py), as show in Fig. 46.
- **Single-column abstraction (SCA):** Specialized Fortran format, in which the horizontal loop dimensions are ignored in scientific routines and re-inserted via source-to-source compiler tools for most efficient execution on CPU or GPU. An example of such source-to-source compiler is CLAW, which can be used to auto-generate optimised Fortran code for CPU and GPU back-ends from a single-column input format (see Fig. 46).

The implementation and conversion of individual components will rely on a mixture of manual ports and automated tool-based conversion of existing source code. The additional tooling infrastructure clearly needs close integration with the existing build infrastructure, so that the various stages of platform-specific optimisation become integrated into the build process.

Gridpoint stencil computations: When, in a routine, data dependencies are only present within each single vertical grid column separately, porting the routine to the SCA is a logical choice. However if the routine employs stencil computations involving horizontal data dependencies between grid columns the SCA breaks down, and a grid-aware DSL is required. An example of a grid-aware DSL, operationally used by MeteoSwiss for COSMO, is GridTools. GridTools currently only supports structured regular grids with i, j-indexing. One can then simply code a *horizontal_diffusion* computation using GridTools, illustrated in Fig. 47.

In the Laplacian stencil computation in this code listing the content within the *Do* scope is referred to as the computational kernel, and is executed for each grid column in a horizontally domain-decomposed structured grid. Loop bounds are implicit knowledge of the *domain*, as well as indexing (i, j horizontally), and data-layout in memory. Additional implicit knowledge of the domain is the *vertical_region* with its loop bounds $kstart, kend$. The second part in this code listing (*horizontal_diffusion*) calls the *laplacian* stencil computation as a function, whose output serves as input for subsequent function calls.

```
stencil laplacian {
    field out, in
    Do {
        vertical_region(kstart,k_end) {
            out = 4*in - (in[i+1] + in[j+1] + in[i-1] + in[j-1])
        }
    }
}
stencil horizontal_diffusion {
    field p_lap, f_flx, p_data, p_fly, p_coeff
    Do{
        laplacian(p_lap, p_data)
        flx(p_flx, p_data, p_lap)
        fly(p_fly, p_data, p_lap)
        out(p_data, p_flx, p_fly, p_coeff)
    }
 }
```

Figure 47: COSMO DSL example of a horizontal diffusion stencil computation for regular structured grids.

The DSL and its optimisation tool-chain are free to restructure or fuse these different computational kernels to gain performance or eliminate costly memory accesses, by introducing temporary caches local to a parallel thread. It can achieve this by carefully analysing data dependencies between kernels by tracking which fields are accessed as read-only and which fields are updated. More importantly, different optimisation choices can be made depending on different target hardware, without introducing extra code in the scientific algorithm. Hardware and parallelisation specific directives (or pragmas) are removed from the scientific algorithms, although it may have well been introduced in the automatically generated code.

Stencil computations typically require halo-exchanges. The DSL tool-chain can analyse when halos are required to be accessed and introduce halo-exchanges appropriately, or may choose to avoid a halo-exchange by performing extra redundant computations in one halo layer previously.

The example (Fig. 47) works well in the case of the COSMO model with its structured grid. IFS however operates on a reduced octahedral Gaussian grid as shown in Fig. 23. Also the ICON model is not using a structured grid but rather the icosahedral grid. One can then not simply use the above syntax with i, j-indexing. IFS-FVM e.g. relies on *Atlas* to access lookup tables that relate neighbouring grid columns. GridTools currently does not support this, but close collaboration between ECMWF and MeteoSwiss / CSCS is leading the way to have *Atlas* and GridTools integrate closely so that access to unstructured grid lookup tables as well as halo-exchanges are arranged via *Atlas*. The aim is to develop a DSL that supports unstructured grids, and hence will be applicable both to IFS and ICON[95] models. The specific DSL syntax is still under development, following an ESCAPE-2 workshop that has provided an important and wide-ranging overview of the community's needs [84].

In this workshop, one particular stencil computation used in IFS-FVM's elliptic solver has been demonstrated, written in Fortran. A possible DSL implementation has been proposed using a yet to be designed syntax. The original Fortran version and the DSL version are shown in appendix (Fig. 54). It is important to note that the notation in this DSL implementation is agnostic to data-layout in memory, and has no OpenMP directives. After application of the DSL-tool-chain optimisations, the auto-generated code is optimal and executable on target hardware, provided that the DSL tool-chain implements a back-end specific to the target hardware. If a new hardware is introduced, a new DSL tool-chain back-end should be developed, without any modifications to the scientific algorithms. If this strategy proves successful (and it already has for MeteoSwiss), we will have achieved a hardware portable, but more importantly performance portable forecasting system.

Single-column abstraction (SCA): The meteorological community as a whole is putting in significant effort to bring modern software engineering techniques to bear on the modelling of atmospheric dynamics. However, modelling of the sub-grid-scale physical processes affecting the state of the atmosphere, which represents around 25% of the run-time cost of a coupled IFS forecast, and around 25% of the lines of code, is not currently seeing the same degree of effort. A number of intertwined reasons can be found to explain this. Firstly, the scale of the problem in terms of numbers of lines of code is substantial, involving many hundreds of thousands of lines of Fortran, some of which have been around for decades. Secondly, the number of scientists working on the parametrizations is large, due to the scientific specialisation required to contribute meaningfully to a given physical process. Finally, the rate of change of lines in the physical parametrization code is higher than in the dynamics. The goal of GPU-readiness for the IFS will in any case require per-file changes in the physics code. Manual insertion of OpenACC statements throughout the code would be one way of achieving this goal, but would lead to a signifi-

[95] Icosahedral Nonhydrostatic

cant additional maintenance burden, as well as many other downsides. Instead, an approach allowing the progressive separation of algorithmic description from software implementation will be followed. A preliminary required stage for future developments is the implementation of an abstraction layer that will allow *Atlas* fields storage to be implemented transparently. At the same time, the single central OpenMP loop over NPROMA blocks in which the physical parametrizations are called, will be split into multiple loops (see code examples in Fig. 51a and Fig. 51b). One OpenMP NPROMA block loop for each physical parametrization will allow proper testing of GPU code, one parametrization at a time. This is not possible with the current single loop in the physics control flow. More generally, the implementation of the abstraction layer will also provide the opportunity to improve code structure in the IFS physics as a whole.

In a second stage, the current parametrizations will be transferred to a SCA. This change should not be overly invasive, as the main algorithmic implementation choices should remain untouched. The targeted SCA relies on the same Fortran code, where all hard-coded horizontal loops have been stripped from the kernels and moved to the driver level (Fig. 53 in the Appendix), and on a source-to-source pre-compilation step which reinserts horizontal loops in an architecture-specific fashion. The performance of the current hand-tuned and hand-optimised code should be retrieved when running on a CPU, and GPU acceleration currently relying on the insertion of OpenACC directives, and a reordering of the driver-level looping. The SCA code should be shorter, with the removal of all horizontal references in the physics kernels. As a consequence, it should also be less confusing and easier to maintain. The code in its SCA-form should compile and execute correctly as is, albeit more slowly due to the poor memory access patterns described by the single-column code. Hardware-specific execution, be it on CPU or on GPU, will require an extra pre-compilation step in the build process, which will generate the optimised code. This optimised architecture-specific code will be easily human-readable, and usable with standard debugging tools. An illustration of the SCA physics kernel changes is provided in code snippet (see Fig. 53 and the proposed control flow changes Fig. 51 in the Appendix).

Further into the future, the physics kernels themselves could be written in a high-level DSL that is bespoke to physical parametrizations. This will achieve:

- a simpler, higher level syntax for the scientists, where only algorithmic requirements are specified;
- vastly improved preliminary assessment tools, and quicker debugging;
- improved flexibility, allowing targeting of multiple architectures from the same algorithmic formulation;
- potentially improved performance on current hardware, due to the optimisations which can be carried out between the DSL and the compiled code-reference.

4.2.4 Co-models and mixed precision

Both co-models, i.e. the concurrent execution of model components, and mixed precision have been explored already in the first phase of the programme, but will be further developed towards operational implementation in the second phase. Both are independent from the algorithmic and programming model research described above, and are expected to produce efficiency gains that are additive to those produced in these areas.

Concurrent execution of model components: Extracting performance from future high-performance computing architectures will mandate the exploitation of parallelism at scales which are orders of mag-

nitude larger than today. However, only a few applications will be able to exploit such extreme levels of parallelism across a single granularity thus requiring that parallelism is exposed in the application across multiple levels. For the IFS, there is potential for increasing parallelism by overlapping computations across different levels in the software stack such as that of components of the Earth-system model as demonstrated in Phase 1 for the radiation scheme and WAM (Section 2.3.1).

At the highest level, Earth-system components such as atmospheric dynamics, atmospheric physics, the wave model, the ocean model, could be executed in parallel as long as data dependencies between them are satisfied. Fig. 48 illustrates the execution sequence of Earth-system components within one time step, for the current IFS in the top view, and for a co-model-based approach which could be implemented in the future, in the lower view.

A possible requirement that a particular component needs to satisfy in order to become a co-model is to have its own grid and perform its own time integration, although the latter is not strictly necessary (e.g., radiation). Having their own grid allows easier independent execution of each co-model on separate, possibly heterogeneous hardware and memory spaces. An example of this would be the radiation processes running on GPUs, with CPUs executing the rest of the Earth-system components. For co-models that perform their own time integration such as the ocean model, this can be executed in parallel with the atmospheric dynamics and physics by using data from the previous time step or by implementing a more advanced implicit time-stepping approach.

Furthermore, whereas currently the radiation, ocean and wave models are only executed every few atmospheric iterations due to restrictions on time-to-solution, a co-model approach removes them from the critical path and allows their execution in lock-step to the atmosphere albeit possibly using smaller time steps. This can lead to improvements to the forecast as well as a significant reduction of the cost of a single time-step since it reduces the critical path to only the atmospheric dynamics and physics as seen in Fig. 48.

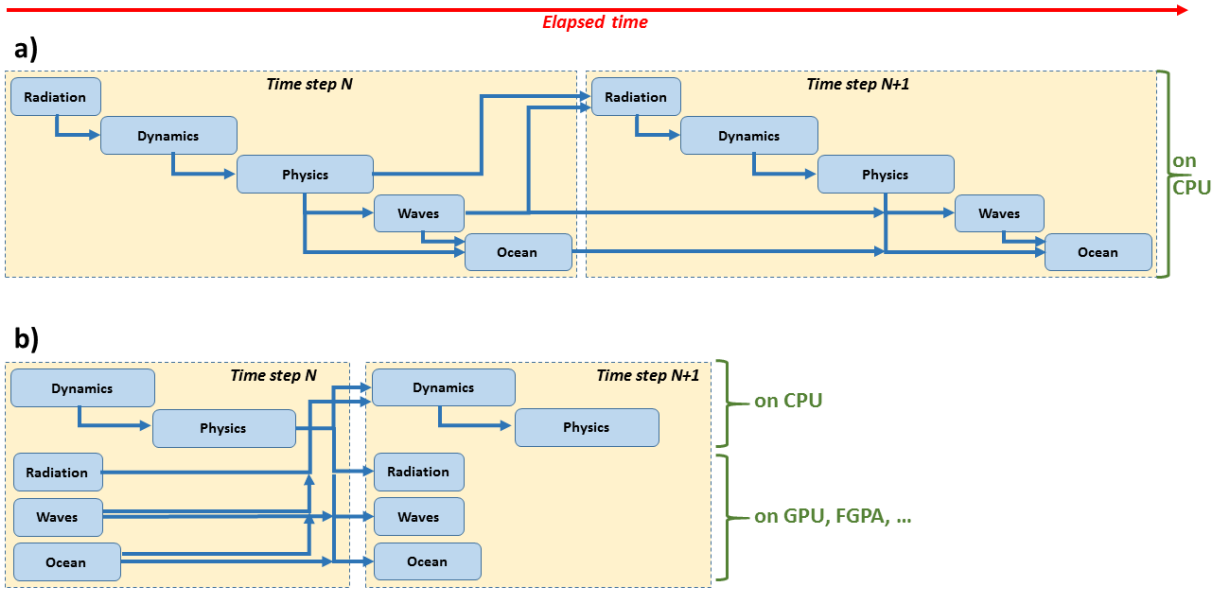


Figure 48: Present IFS execution sequence on CPU (a) and future co-execution of model components on CPU and other processor types (b); see also Section 2.3.1.

The co-model approach also brings further advantages when it comes to porting code to different architectures since it breaks up the current monolithic implementation into smaller, self-contained individual components. Here, the dwarf concept - pioneered by the Scalability Programme in ESCAPE - pays off. This is important since certain components exhibit computational patterns that map well to conventional multi-core processors while others to many-core accelerators. As a result, not only is porting easier since the size of the code-base is reduced significantly but potentially more efficient since only the co-models that are suitable for accelerators or other architectures are ported.

Mixed precision: Initially, the use of single precision is limited to the forecast model and tests to apply single precision in data assimilation are still at the very beginning. While reduced precision was used successfully for data assimilation with an EnKF [55], the application of single precision within 4D-Var will be more difficult. Here, the TL [96] and AD [97] realisations need to be consistent to allow for convergence of the conjugate gradient solver that is used for minimisation. Experience shows that the standard test for ensuring that the tangent-linear and adjoint models conform must be passed by at least 10 decimal digits of precision. This would not be possible when using single precision. However, a recent study of reduced precision based on the OOPS framework that applied 4D-Var data assimilation to a quasi-geostropic model has shown very promising results. Precision could be reduced all the way to half precision if re-orthogonalisation was applied to the residuals of the conjugate gradient algorithm.

Notwithstanding, one should not underestimate however the effort required today to maintain the TL/AD model as close as possible to the non-linear model to assure convergence of the evolution of small perturbations. This is especially true where not the entire trajectory is stored but important trajectory elements are recomputed in the TL/AD model, and thus a lower precision TL/AD approximation may make this task much more difficult. At the same time, there is a trend towards direct assimilation and monitoring of space-borne cloud radar and lidar observations, including the assimilation of rain and lightning, which likely increases the importance and complexity of associated TL/AD approximations of moist physical processes.

Nevertheless, for selected parts it might be possible to use a level of numerical precision below single precision within IFS in the future. This is in particular interesting since the boom of deep learning algorithms is driving recent hardware developments towards the acceleration of linear algebra at reduced reduced numerical precision with 16 bits or less to represent real numbers. It is unlikely that this level of precision can be exposed to the entire IFS. However, it may be possible to accelerate expensive model components or computing kernels. [53] could, for example, show that the TensorCore accelerator on Volta GPUs of NVIDIA could potentially be used to calculate the Legendre transformations of IFS at half precision with no significant impact on forecast results. A reduction of numerical precision below single precision may also be possible for some of the parametrization schemes [94].

4.3 Machine learning

New methods of machine learning such as deep neural networks allow to learn and emulate the dynamics of complex non-linear systems. Machine learning tools can also be used to extract information from complex data sets and to understand the connectivity between model parameters in non-linear environments. Due to this capability, there are numerous applications for these methods when studying the Earth system due to the non-linear character of the different Earth-system components, measurements that are

[96] Tangent Linear
[97] Adjoint model

often under-sampled or perturbed, and since models and emulators that are based on physical principles often rely on rough approximations.

Furthermore, the similarities between machine learning and data assimilation are well-known [58], and data assimilation has even potential to derive information on the governing differential equations of the atmosphere through machine learning [11]. The new machine learning tools also promise to run efficiently on modern supercomputers, to be easily portable to heterogeneous hardware due to the availability of efficient machine learning libraries and there is plenty of Earth-system data available for training (210 PB of primary data at ECMWF).

There are therefore a very large number of potential applications of the new machine learning tools within the workflow of ECMWF that include the use of new sources of observations, observation pre-processing and bias corrections, data assimilation, numerical weather predictions, and data post-processing.

Aspects of machine learning are well established in numerical weather prediction (e.g. learning model error in weak constraint 4D-Var, neural network products for soil moisture, fast observation operators, some fast physics calculations such as radiation). Of interest is to what extent new techniques from machine learning developed in applications such as image or speech recognition can be projected into the domain of weather and climate science. It is unlikely that machine learning will replace numerical model formulations that are based on basic physics due to the strong non-linearity and large number of degrees of freedom of the Earth system.

Further, with data assimilation, numerical weather prediction already relies on a rigorously-based and highly developed methodology for combining observations and models. It will therefore require research based on scientific principles to identify areas in Earth-system science where the new methods will truly make a difference in the future, and to avoid simply following the current hype of machine learning techniques. It will be the responsibility of the Scalability Programme to identify and exploit synergies between applications of machine learning tools in the different parts of the ECMWF workflow, to support the investigation of common scientific questions, and to build common infrastructure for applications.

The main focus of the Scalability Programme in short-term research will be:

- on the development and testing of neural network emulators for model components and in particular physical parametrization schemes. While first results are promising, this tasks is of *high risk* since it has yet to be shown whether for example deep neural networks will be able to represent the complexity of the underlying physical schemes of the operational forecast model sufficiently. The task will also require to answer some difficult scientific questions, for example how to enforce conservation laws or how to use knowledge about the underlying physical system to improve the design of neural networks. However, the task may also generate *high gains* with large speed-up factors for model performance. The work can be based on the work to emulate the radiation scheme.

- on the use of deep learning emulators to generate tangent linear and adjoint model code. Since the arithmetic of neural networks is simple and repetitive, the generation of tangent linear and adjoint code would likely be much simpler for neural networks when compared to the same code generated for the conventional forecast model. This task is of *high risk* since it will not only rely on the results of the previous task, it will also require more work to design neural networks (in particular regarding activation functions and communication patterns) that are best suited for the derivation of tangent linear and adjoint code. While there are automated tools available in common machine learning libraries to generate such code automatically, automated tools are unlikely to provide sufficient performance. However, the task is also of *high gain* since it may not only allow

for significant speed-ups of data-assimilation, it may also allow to reduce the effort of scientists to generate tangent linear and adjoint code for complex parametrization schemes. A task which is in itself so complicated that large parts of the forecast model code can only be transformed in simplified versions and some of the model components, such as the radiation scheme, are currently still based on outdated versions of the forward model. The existing neural network emulator for the radiation scheme could be used to start with tests very soon.

- on the use of neural networks to enhance ensemble predictions. In particular, to train neural networks that are using model output of a number of ensemble members as input to generate ensemble spread and ensemble mean fields that are as meaningful as the equivalent fields from ensembles with a larger number of ensemble members. This task is of *high risk* since it is yet unknown whether the neural networks will be able to pick up the underlying non-linear dynamics in sufficient detail. However, it is also of *high gain* since it may allow to reduce the number of ensemble members in predictions to reduce computational cost significantly.

5 The second phase: Implementation

At the beginning of Phase 2 of the programme, selected topics need special attention for resource planning:

- The implementation of the *Atlas* library throughout the IFS is crucial for success. As this will be intrusive, it will be performed in three steps, initially as an added layer linking to existing data structures, then as a full implementation replacing existing data structures, and lastly interfacing with the DSL tool-chain. In the future, the maintenance of *Atlas* as an open-source code and its (currently optional) link with GridTools needs to be planned.

- The coordination between the GPU developments for the IFS-ST and the tangent-linear and adjoint versions of the code used in data assimilation require special and as yet unexplored care. The same applies to the coupled components. With atmospheric chemistry further work on load balancing, memory efficiency and concurrent execution is mandatory.

- For the coupled model components like NEMO and SI3[98], infrastructure developments depend on consortia and can not be planned independently.

5.1 Efficiency gains

From the existing research performed in Phase 1 of the programme there is strong evidence that the forecasting system will be more efficient in terms of both time and energy to solution for code execution, and task execution along the workflow. As mentioned several times, a key to obtaining efficiency gains is building flexibility into the system so that configuration options can be compared to one another and scientific (forecast quality) and computing/data handling performance can be traded off against each other. This is important as ECMWF is running the IFS across different applications for its Member States and Copernicus services, with a wide range of setups and forecast ranges in high resolution and as ensembles of analyses and forecasts.

[98] Sea Ice modelling Integrated Initiative

An important aspect of efficiency gains is the simplicity to learn how the system works, to manage, modify and maintain code infrastructures and to create benchmarks for future procurements, with testing on a variety of external HPC infrastructures. The present system is complex and difficult to maintain as a single code base with emerging HPC choices. In the future, performance portable codes and data-centric workflows are based on key building blocks that allow to separate science from hardware concerns. Any type of data is managed through object based datastores and I/O servers either in the cloud or at home. Such changes will offer an entirely different way of producing, accessing and processing (or learning) forecast data. Quantifying this type of efficiency is probably impossible.

A modular forecasting system will be more resilient and more portable. This has been already demonstrated for observational data processing and through the operational implementation of FDB5, but also applies to the IFS and OOPS once the proposed developments become available. Again, quantifying the effect of enhanced resilience on efficiency is difficult at this time.

In terms of achievable and quantifiable performance gains, the following estimate is made:

- *Algorithms:* The investment in new numerical and assimilation methodologies has already shown scalability advances. For the model, semi-implicit time-stepping schemes continue to be important as they allow for the use of larger time-steps and therefore deliver time-to-solution performance. Improvements to the solution of 3D elliptic solvers for grid-point models such as in IFS-FVM include further work on general optimisations, pipelining (overlapping instructions), pre-conditioning and the use of multi-grid methods. The medium-term target for IFS-FVM is to be of comparable cost as the IFS-ST, which means a further acceleration by at least a factor of 2.
  With the hybrid 4D-Var formulation, there is still room to stretch the overall minimisation cost across a longer time window, towards a quasi-continuous assimilation. This will reduce maximum node but not node-hour allocation.
  Machine learning could deliver 10-fold gains for individual model components but it is as yet unclear how this will translate into accelerating the overall performance, notwithstanding the training effort required. If the entire model physics can be accelerated by a factor of 10, the model would become about 20% faster.

- *Processor technology and programming:* The short-term implementation of mixed precision, concurrent execution of model components, and improved overlapping of computation will, at best, achieve an improved time to solution by a factor of 2. Using lower than single precision, for example in the spectral transforms, may produce additional gains that are difficult to predict now. However, these developments carry the additional benefit to be applicable on future processors specifically designed for lower precision executions. Upgrades in present-day CPU technology, better single-CPU node performance with more cores, and increased memory and memory bandwidth are expected to provide another factor of 1.5, i.e. an overall improvement by a factor 3 with present codes and technology.
  From the tests with dwarfs on GPUs individual acceleration factors of 20 or more when allocated across multiple nodes with high-speed interconnects were achieved. If this is possible for the spectral transforms, the advection, and the physical parametrizations, a substantial part of the model would execute on the GPUs, and the overall coupled model cost could be reduced by at least a factor of 2.5. If adapting both WAM and NEMO to GPUs and optimising the execution and coupling of all components across CPUs and GPUs correspondingly reduces their cost, an overall gain of a factor of 5 may be obtained. Using GPU, FPGA and neural networks, the latter for physics parametrizations, will further reduce energy-to-solution beyond the speed-up, thus increase value for money.
  The work on DSL tool-chains mostly aims at performance portability and is an investment in af-

fordability, energy efficiency and the long-term sustainability of the IFS. The analysis system will directly benefit from the forecast model efficiency gains, and its scalability is mostly determined by the scalability of the model. The non-linear model trajectories produce about 45% of the overall cost of 4D-Var. However, it is essential to adapt the tangent-linear and adjoint model versions to GPUs for achieving a similar overall gain as for the non-linear forecast model. In coupled data assimilation, the adaptation of NEMO and WAM are equally important. In addition, it is expected to obtain a 20% speed-up from introducing OOPS.

- *Data handling and data flow*: The cost of observational data pre-processing and screening including the application of radiance observation operators has been reduced already significantly in the first phase. The introduction of the observation object datastore is expected to reduce cost further and add the performance necessary for dealing with more hyper-spectral satellite sounders and IoT observations.
  FDB5 already achieved acceleration of product generation by a factor of 5, which is expected to be doubled once product generation can run on the fly on non-volatile memory extensions. The MultIO I/O server will allow to hide I/O cost for all model components, which is important at higher resolutions.

5.2 Resources

In the past five years, significant resources have been invested in the Scalability Programme. ECMWF's Council assigned five core staff positions to the programme that have been invested in RD (Earth-system modelling, assimilation and IFS sections) and FD (Software development section). ECMWF Members States also devoted staff to the development of OOPS and the present IFS I/O server. In addition, twelve staff members are funded by external projects, and there are in-kind provisions of effort for coordination and management, and partly other, related technical work.

A key challenge will be to sustain this level of support and effort. This is a key requirement for the success of the programme and for the successful transition of the various research efforts into operations and in order to have available a benchmark for the 2024 procurement that returns the best possible value for money. The same applies for the workflow topic, which will enable ECMWF to provide the best possible service to Member States and provide the Copernicus services with a future-proof infrastructure.

Of particular concern is the external funding. This contributes the largest part to the resources, is highly competitive, and is based on opportunity driven projects which are not intrinsically targeting ECMWF concerns. Most of them originate from the European Commission's Future and Emerging Technology programme that is now managed by the EuroHPC Joint Undertaking. As EuroHPC relies on co-funding, ECMWF will only be able to claim 50% of the total cost, which means that - even with a similar proposal success rate as in the past - ECMWF will automatically loose half of the above mentioned resources.

It is therefore of utmost importance for ECMWF to define an appropriate resourcing strategy to successfully complete the programme. The work needs to be coordinated across ECMWF departments, with Member States and with collaboration partners. Particularly, all Member States who are sharing code with ECMWF need to be involved in the roadmap implementation and resource planning process.

6 Conclusions and Outlook

This paper has compiled the significant research and development efforts invested by ECMWF into the preparation of the forecasting system for the future challenges of super-computing and the handling of very large amounts and diversities of data. ECMWF recognised the associated challenges early and founded its Scalability Programme that has reached half-time. The programme delivered many outstanding results that are already starting to produce benefits in the operational system today. Through this programme, ECMWF has assumed a scientific leadership in this new discipline sitting between domain-specific applied and computational science. This achievement would not have been possible without the commitment by Member States with own resources and by allowing ECMWF to allocate staff to this effort.

Most importantly, the programme puts ECMWF in a position where it can plan the full implementation of these novel developments into operational production with a solid scientific foundation, and without compromising other relevant scientific developments. It is important for ECMWF to ensure that this implementation is sufficiently supported through coordination with all relevant partners as well as through adequate financial and human resources. With this, it is our firm belief that an overall efficiency gain factor of 15 (a factor 3 upgrade from CPU technology progress and present-day code enhancements, and a factor 5 from the inclusion of GPUs) beyond the performance of the operational system in 2019 is achievable within the next 8 years, which is essentially continuing the historical compute performance gain of ECMWF's IFS, closely associated with Moore's law, but through the combined efforts of the Scalability Programme.

References

[1] S. Adams, R. Ford, M. Hambley, J. Hobson, I. Kavcic, C. Maynard, T. Melvin, E. Müller, S. Mullerworth, A. Porter, M. Rezny, B. Shipway, and R. Wong. LFRic: Meeting the challenges of scalability and performance portability in weather and climate models. *Journal of Parallel and Distributed Computing*, 132:383 – 396, 2019.

[2] M. Alvanos and T. Christoudias. GPU-accelerated atmospheric chemical kinetics in the ECHAM/MESSy (EMAC) earth system model (version 2.52). *Geoscientific Model Development*, 10(10):3679–3693, 2017.

[3] G. M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, AFIPS '67 (Spring), pages 483–485, New York, NY, USA, 1967. ACM.

[4] ARM. ARM MAP. https://www.arm.com/products/development-tools/server-and-hpc/forge/map.

[5] K. Asanovic, R. Bodik, B. C. Catanzaro, J. J. Gebis, P. Husbands, K. Keutzer, D. A. Patterson, W. L. Plishker, J. Shalf, S. W. Williams, and K. A. Yelick. The landscape of parallel computing research: a view from Berkeley. Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, Dec 2006.

[6] K. Asanovic, J. Wawrzynek, D. Wessel, K. Yelick, R. Bodik, J. Demmel, T. Keaveny, K. Keutzer, J. Kubiatowicz, N. Morgan, D. Patterson, and K. Sen. A view of the parallel computing landscape. *Communications of the ACM*, 52(10):56, oct 2009.

[7] T. Auligné, B. Ménétrier, A. C. Lorenc, and M. Buehner. Ensemble–variational integrated localized data assimilation. *Monthly Weather Review*, 144(10):3677–3696, 2016.

[8] P. Bauer, M. C. Morgan, and S. Sbill. Extreme-scale computing and data handling - the heart of progress in weather and climate prediction. *WMO Bulletin n°*, 68:1, 2019.

[9] T. Ben-Nun, J. de Fine Licht, A. N. Ziogas, T. Schneider, and T. Hoefler. Stateful dataflow multigraphs: A data-centric model for high-performance parallel programs. *CoRR*, abs/1902.10345, 2019.

[10] É. Blayo, M. Bocquet, E. Cosme, and L. F. Cugliandolo. *Advanced Data Assimilation for Geosciences: Lecture Notes of the Les Houches School of Physics: Special Issue, June 2012*. Oxford University Press, USA, 2014.

[11] M. Bocquet, J. Brajard, A. Carrassi, and L. Bertino. Data assimilation as a deep learning tool to infer ODE representations of dynamical models. *Nonlin. Processes Geophys.*, 2019.

[12] M. Bocquet, L. Wu, and F. Chevallier. Bayesian design of control space for optimal assimilation of observations. part i: Consistent multiscale formalism. *Quarterly Journal of the Royal Meteorological Society*, 137(658):1340–1356, 2011.

[13] M. Bonavita, Y. Trémolet, E. Hólm, S. Lang, M. Chrust, M. Janisková, P. Lopez, P. Laloyaux, P. de Rosnay, M. Fisher, M. Hamrud, and S. English. A strategy for data assimilation, 2017.

[14] S. Borkar and A. A. Chien. The future of microprocessors. *Communications of the ACM*, 54(5):67–77, 2011.

[15] N. Bousserez and D. K. Henze. Optimal and scalable methods to approximate the solutions of large-scale bayesian problems: theory and application to atmospheric inversion and data assimilation. *Quarterly Journal of the Royal Meteorological Society*, 144(711):365–390, 2018.

[16] G. Brunet, S. Jones, P. M. Ruti, et al. *Seamless prediction of the Earth System: from minutes to months*. World Meteorological Organization, 2015.

[17] R. Buizza, M. Alonso-Balmaseda, A. Brown, S. English, R. Forbes, A. J. Geer, T. Haiden, M. Leutbecher, L. Magnusson, M. Rodwell, M. Sleigh, T. Stockdale, F. Vitart, and N. Wedi. The development and evaluation process followed at ECMWF to upgrade the integrated forecasting system (IFS). *ECMWF Technical Memoranda*, (829), 10 2018.

[18] B. S. Center. BSC performance tools. https://www.bsc.es/discover-bsc/organisation/scientific-structure/performance-tools. Accessed: 2019-07-18.

[19] B. S. Center. Scalasca. https://www.scalasca.org. Accessed: 2019-07-18.

[20] F. Chevallier, F. Chéruy, N. A. Scott, and A. Chédin. A neural network approach for a fast and accurate computation of a longwave radiative budget. *Journal of Applied Meteorology*, 37(11):1385–1397, 1998.

[21] S. Choi and S. Hong. A global non-hydrostatic dynamical core using the spectral element method on a cubed-sphere grid. *Asia-Pacific Journal of Atmospheric Sciences*, 52:291–307, 2016.

[22] V. Clement, S. Ferrachat, O. Fuhrer, X. Lapillonne, C. E. Osuna, R. Pincus, J. Rood, and W. Sawyer. The CLAW DSL: Abstractions for performance portable weather and climate models. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, PASC '18, pages 2:1–2:10, New York, NY, USA, 2018. ACM.

[23] P. Colella. Defining software requirements for scientific computing. DARPA HPCS Presentation, 2004.

[24] F. Cooper, P. D. Dueben, C. Denis, A. Dawson, and P. Ashwin. It may be possible to reduce the amount of data output by weather forecast models by reducing numerical precision with lead time. *accepted in Monthly Weather Review*, 2019.

[25] Cray. Cray performance measurement and analysis. https://pubs.cray.com/content/S-2376/CrayPat

[26] W. Deconinck, P. Bauer, M. Diamantakis, M. Hamrud, C. Kühnlein, P. Maciel, G. Mengaldo, T. Quintino, B. Raoult, P. K. Smolarkiewicz, et al. Atlas: A library for numerical weather prediction and climate modelling. *Computer Physics Communications*, 220:188–204, 2017.

[27] R. H. Dennard, F. H. Gaensslen, V. L. Rideout, E. Bassous, and A. R. LeBlanc. Design of ion-implanted MOSFET's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, 9(5):256–268, 1974.

[28] L. Douriez, A. Gray, D. Guibert, P. Messmer, and E. Raffin. Performance report and optimized implementations of Weather & Climate dwarfs on multi-node systems. Technical report, ESCAPE, 2018.

[29] J. Doyle. Overview of the Navy's Next-Generation NEPTUNE Modeling System. AMS, 2018. 25th Conference on Numerical Weather Prediction, June 04-08, 2018, Denver, CO.

[30] P. Dueben and M. Acosta. Reduced numerical precision in atmosphere models and computational performance evaluation. 2019.

[31] P. Dueben, P. Bauer, J.-N. Thepaut, V.-H. Peuch, A. Geer, and S. English. Machine learning at ECMWF. *Research Department Memorandum*, 2019.

[32] P. D. Dueben and P. Bauer. Challenges and design choices for global weather and climate models based on machine learning. *Geoscientific Model Development*, 11(10):3999–4009, 2018.

[33] P. D. Dueben, M. Leutbecher, and P. Bauer. New methods for data storage of model output from ensemble simulations. *Monthly Weather Review*, 147(2):677–689, 2019.

[34] P. D. Dueben, A. Subramanian, A. Dawson, and T. Palmer. A study of reduced numerical precision to make superparameterization more competitive using a hardware emulator in the OpenIFS model. *Journal of Advances in Modeling Earth Systems*, 9(1):566–584, 2017.

[35] ECMWF. ECMWF Strategy 2015-2025 - the strength of a common goal. 2015.

[36] ECMWF. *ecProf meets the High Resolution IFS Forecast model at ECMWF*, 2016. 17th HPC Workshop in Meteorology.

[37] H. C. Edwards and D. Sunderland. Kokkos array performance-portable manycore programming model. In *Proceedings of the 2012 International Workshop on Programming Models and Applications for Multicores and Manycores*, pages 1–10. ACM, 2012.

[38] H. C. Edwards, C. R. Trott, and D. Sunderland. Kokkos: Enabling manycore performance portability through polymorphic memory access patterns. *Journal of Parallel and Distributed Computing*, 74(12):3202 – 3216, 2014. Domain-Specific Languages and High-Level Frameworks for High-Performance Computing.

[39] A. L. C. Facility. Darshan. https://www.mcs.anl.gov/research/projects/darshan/.

[40] M. Fisher. Development of a simplified kalman filter, 1998.

[41] M. Fisher and S. Gürol. Parallelization in the time dimension of four-dimensional variational data assimilation. *Quarterly Journal of the Royal Meteorological Society*, 143(703):1136–1147, 2017.

[42] M. Fisher, Y. Tremolet, H. Auvinen, D. Tan, and P. Poli. Weak-constraint and long-window 4D-Var. *Techn. Rep*, 655, 2011.

[43] A. Fouilloux. ODB (observational database) and its usage at ecmwf. In *Twelfth Workshop on Meteorological Operational Systems, 2-6 November 2009*, pages 86–90, Shinfield Park, Reading, 2009. ECMWF, ECMWF.

[44] E. Fucile, C. Zanna, I. Mallas, M. Crepulja, M. Suttie, and D. Vasiljevic. SAPP: a new scalable acquisition and pre-processing system at ECMWF. *ECMWF Newsletter*, pages 37–41, 2014.

[45] O. Fuhrer, T. Chadha, T. Hoefler, G. Kwasniewski, X. Lapillonne, D. Leutwyler, D. Lüthi, C. Osuna, C. Schär, T. C. Schulthess, and H. Vogt. Near-global climate simulation at 1 km resolution: establishing a performance baseline on 4888 GPUs with COSMO 5.0. *Geosci. Model Dev.*, 11(4):1665–1681, 2018.

[46] G. H. Golub and C. F. Van Loan. Matrix computations. *The Johns Hopkins University Press, Baltimore, USA*, 1989.

[47] S. Gratton, S. Gürol, E. Simon, and P. L. Toint. Guaranteeing the convergence of the saddle formulation for weakly constrained 4d-var data assimilation. *Quarterly Journal of the Royal Meteorological Society*, 144(717):2592–2602, 2018.

[48] S. Gratton, S. Gürol, E. Simon, and P. L. Toint. A note on preconditioning weighted linear least-squares, with consequences for weakly constrained variational data assimilation. *Quarterly Journal of the Royal Meteorological Society*, 144(712):934–940, 2018.

[49] T. Gysi, C. Osuna, O. Fuhrer, M. Bianco, and T. C. Schulthess. STELLA: A domain-specific tool for structured grid methods in weather and climate models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '15, pages 41:1–41:12, New York, NY, USA, 2015. ACM.

[50] T. Gysi, C. Osuna, O. Fuhrer, M. Bianco, and T. C. Schulthess. Stella: A domain-specific tool for structured grid methods in weather and climate models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, page 41. ACM, 2015.

[51] N. Halko, P.-G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[52] J. Haseler. *Early-delivery suite*. European Centre for Medium-Range Weather Forecasts, 2004.

[53] S. Hatfield. Reduced-precision arithmetic in numerical weather prediction with an emphasis on data assimilation. *DPhil Thesis at the University of Oxford, submitted*, 2019.

[54] S. Hatfield, M. Chantry, P. Dueben, and T. Palmer. Accelerating high-resolution weather models with deep-learning hardware. *Best Paper Award in Proceedings of PASC2019*, 2019.

[55] S. Hatfield, P. Düben, M. Chantry, K. Kondo, T. Miyoshi, and T. Palmer. Choosing the optimal numerical precision for data assimilation in the presence of model error. *Journal of Advances in Modeling Earth Systems*, 10(9):2177–2191, 2018.

[56] M. D. Hill and M. R. Marty. Amdahl's law in the multicore era. *Computer*, 41(7):33–38, July 2008.

[57] S. Hong, Y. Kwon, T. Kim, and et al. The Korean Integrated Model (KIM) System for Global Weather Forecasting. *Asia-Pacific Journal of Atmospheric Sciences*, 54:267–292, 2018.

[58] H. Hsieh and B. Tang. Applying neural network models to prediction and data analysis in meteorology and oceanography. *Bulletin of the American Meteorological Society*, 79(9):1855–1870, 1998.

[59] A. Hunt. *The pragmatic programmer*. Pearson Education India, 1900.

[60] Intel. Intel VTune Amplifier. https://software.intel.com/en-us/vtune.

[61] H. Järvinen, J.-N. Thépaut, and P. Courtier. Quasi-continuous variational data assimilation. *Quarterly Journal of the Royal Meteorological Society*, 122(530):515–534, 1996.

[62] C. A. Keller and M. J. Evans. Application of random forest regression to the calculation of gas-phase chemistry within the GEOS-Chem chemistry model v10. *Geoscientific Model Development*, 12(3):1209–1225, 2019.

[63] P. Kogge and J. Shalf. Exascale computing trends: Adjusting to the 'new normal' for computer architecture. *Computing in Science & Engineering*, 15(6):16–26, 2013.

[64] N. V. Koldunov, V. Aizinger, N. Rakowsky, P. Scholz, D. Sidorenko, S. Danilov, and T. Jung. Scalability and some optimization of the Finite-volumE Sea ice-Ocean Model, Version 2.0 (FESOM2). *Geoscientific Model Development*, 12(9):3991–4012, 2019.

[65] C. Kühnlein, W. Deconinck, R. Klein, S. Malardel, Z. P. Piotrowski, P. K. Smolarkiewicz, J. Szmelter, and N. P. Wedi. FVM 1.0: a nonhydrostatic finite-volume dynamical core for the IFS. *Geoscientific Model Development*, 12(2):651–676, 2019.

[66] C. Kühnlein and P. K. Smolarkiewicz. An unstructured-mesh finite-volume MPDATA for compressible atmospheric dynamics. *Journal of Computational Physics*, 334:16–30, 2017.

[67] C. Kühnlein and P. K. Smolarkiewicz. A nonhydrostatic finite-volume option for the IFS. *ECMWF Newsletter*, 158:30–36, 2019.

[68] P. Laloyaux, M. Balmaseda, D. Dee, K. Mogensen, and P. Janssen. A coupled data assimilation system for climate reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 142(694):65–78, 2016.

[69] P. Lean, M. Bonavita, E. Hólm, N. Bormann, and T. McNally. Continuous data assimilation for the IFS, 2019.

[70] A. C. Lorenc. The potential of the ensemble kalman filter for NWP - a comparison with 4D-Var. *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography*, 129(595):3183–3203, 2003.

[71] A. C. Lorenc and M. Jardak. A comparison of hybrid variational data assimilation methods for global NWP. *Quarterly Journal of the Royal Meteorological Society*, 144(717):2748–2760, 2018.

[72] P. Maciel, T. Quintino, B. Raoult, and S. Siemen. MIR - ECMWF's new interpolation package. 2015.

[73] S. Malardel, N. Wedi, W. Deconinck, M. Diamantakis, C. Kühnlein, G. Mozdzynski, M. Hamrud, and P. Smolarkiewicz. A new grid for the IFS. *ECMWF Newsletter*, 146:23–28, 2016.

[74] S. McIntosh-Smith, J. Price, T. Deakin, and A. Poenaru. Comparative benchmarking of the first generation of HPC-optimised ARM processors on Isambard. *Cray User Group*, 5, 2018.

[75] O. E. B. Messer, E. D'Azevedo, J. Hill, W. Joubert, M. Berrill, and C. Zimmer. MiniApps derived from production HPC applications using multiple programing models. *Int. J. High Performance Computing Applications*, 32(4):582–593, sep 2016.

[76] J. Michalakes and A. Reinecke. Developing NEPTUNE for U.S. Naval Weather Prediction. ECMWF, 2018. 18th ECMWF Workshop on HPC in Meteorology, 25 Sept. 2018.

[77] A. Moore and R. Wilson. *FPGAs for Dummies*. Wiley, 2017.

[78] G. E. Moore. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86(1):82–85, 1998.

[79] S. K. Moore. Multicore is bad news for supercomputers. *IEEE Spectrum*, 45(11):15–15, November 2008.

[80] G. Mozdzynski, P. Bauer, and P. Dueben. Report outlining a strategic approach for efficiency savings based on concurrency and accuracy., 2018.

[81] G. Mozdzynski, M. Hamrud, N. Wedi, J. Doleschal, and H. Richardson. A PGAS implementation by co-design of the ECMWF Integrated Forecasting System (IFS). In *2012 SC Companion: High Performance Computing, Networking Storage and Analysis*, pages 652–661. IEEE, 2012.

[82] A. Müller, W. Deconinck, C. Kühnlein, G. Mengaldo, M. Lange, N. Wedi, P. Bauer, P. K. Smolarkiewicz, M. Diamantakis, S.-J. Lock, M. Hamrud, S. Saarinen, G. Mozdzynski, D. Thiemert, M. Glinton, P. Bénard, F. Voitus, C. Colavolpe, P. Marguinaud, Y. Zheng, J. Van Bever, D. Degrauwe, G. Smet, P. Termonia, K. P. Nielsen, B. H. Sass, J. W. Poulsen, P. Berg, C. Osuna, O. Fuhrer, V. Clement, M. Baldauf, M. Gillard, J. Szmelter, E. O'Brien, A. McKinstry, O. Robinson, P. Shukla, M. Lysaght, M. Kulczewski, M. Ciznicki, W. Piątek, S. Ciesielski, M. Błażewicz, K. Kurowski, M. Procyk, P. Spychala, B. Bosak, Z. P. Piotrowski, A. Wyszogrodzki, E. Raffin, C. Mazauric, D. Guibert, L. Douriez, X. Vigouroux, A. Gray, P. Messmer, A. J. Macfaden, and N. New. The ESCAPE project: Energy-efficient scalable algorithms for weather prediction at exascale. *Geoscientific Model Development*, 12(10):4425–4441, 2019.

[83] C. Osuna. Report on the performance portability demonstrated for the relevant weather & climate dwarfs. Technical report, ESCAPE, 2018.

[84] C. Osuna, J. Behrens, R. Budich, W. Deconinck, J. Duras, I. Epicoco, O. Fuhrer, C. Kühnlein, L. Linardakis, T. Wicky, and N. Wedi. D2.1: High-level Domain Specific Language (DSL) specification. Technical report, ECMWF, 2019.

[85] D. Patterson. The trouble with multi-core. *IEEE Spectr.*, 47(7):28–32, July 2010.

[86] D. A. Patterson and J. L. Hennessy. *Computer organization and design: the hardware/software interface*. Newnes, 2013.

[87] C. Pires, R. Vautard, and O. Talagrand. On extending the limits of variational assimilation in nonlinear chaotic systems. *Tellus A*, 48(1):96–121, 1996.

[88] J. W. Poulsen and P. Berg. Tuning the implementation of the radiation scheme ACRANEB2. Technical report, DMI report 17-22, 2017.

[89] J. Ragan-Kelley, C. Barnes, A. Adams, S. Paris, F. Durand, and S. Amarasinghe. Halide: A language and compiler for optimizing parallelism, locality, and recomputation in image processing pipelines. In *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '13, pages 519–530, New York, NY, USA, 2013. ACM.

[90] I. Z. Reguly, G. R. Mudalige, and M. B. Giles. Loop tiling in large-scale stencil codes at run-time with OPS. *IEEE Transactions on Parallel and Distributed Systems*, 29(4):873–886, April 2018.

[91] P. E. Ross. Why CPU frequency stalled. *IEEE Spectr.*, 45(4):72–72, Apr. 2008.

[92] K. Rupp. CPU, GPU and MIC hardware characteristics over time. *URl: https://www. karlrupp. net/2013/06/cpu-gpu-and-mic-hardwarecharacteristics-over-time*, 2013.

[93] S. Saarinen and I. Miller. IFS RAPS18 benchmark - update 2.1. Technical report, ECMWF Technical Notes, 2018.

[94] L. Saffin, S. Hatfield, P. Dueben, and T. Palmer. Using stochastic physics to determine the numerical precision required for the parametrization schemes of a global atmospheric model. *submitted to Q. J. Royal Meteorol. Soc.*, 2019.

[95] T. C. Schulthess. Programming revisited. *Nature Physics*, 11(5):369, 2015.

[96] T. C. Schulthess, P. Bauer, N. Wedi, O. Fuhrer, T. Hoefler, and C. Schär. Reflecting on the goal and baseline for exascale computing: A roadmap based on weather and climate simulations. *Computing in Science & Engineering*, 21(1):30–41, 2018.

[97] P. K. Smolarkiewicz, W. Deconinck, M. Hamrud, C. Kühnlein, G. Mozdzynski, J. Szmelter, and N. P. Wedi. A finite-volume module for simulating global all-scale atmospheric flows. *Journal of Computational Physics*, 314:287–304, 2016.

[98] J. Targett, M. Lange, and O. Marsden. Investigation of FPGA implementation of operational cloud micro-physics parameterisation. 2019. in preparation.

[99] O. Tintó Prims, M. C. Acosta, A. M. Moore, M. Castrillo, K. Serradell, A. Cortés, and F. J. Doblas-Reyes. How to use mixed precision in ocean models: exploring a potential reduction of numerical precision in nemo 4.0 and roms 3.6. *Geoscientific Model Development*, 12(7):3135–3148, 2019.

[100] Y. Trémolet, A. Hofstadler, and W. Deconinck. OOPS as a common framework for research and operations. 2013.

[101] J. Tshimanga, S. Gratton, A. Weaver, and A. Sartenaer. Limited-memory preconditioners, with application to incremental four-dimensional variational data assimilation. *Quarterly Journal of the Royal Meteorological Society*, 134(632):751–769, 2008.

[102] G. Tumolo and L. Bonaventura. A semi-implicit, semi-Lagrangian, DG framework for adaptive numerical weather prediction. *Quarterly Journal of the Royal Meteorological Society*, 141:2582–2601, 2015.

[103] G. Tumolo, L. Bonaventura, and M. Restelli. A semi-implicit, semi-Lagrangian, p-adaptive discontinuous Galerkin method for the shallow water equations. *Journal of Computational Physics*, 232:46–67, January 2013.

[104] P. A. Ullrich, C. Jablonowski, J. Kent, P. H. Lauritzen, R. Nair, K. A. Reed, C. M. Zarzycki, D. M. Hall, D. Dazlich, R. Heikes, C. Konor, D. Randall, T. Dubos, Y. Meurdesoif, X. Chen, L. Harris, C. Kühnlein, V. Lee, A. Qaddouri, C. Girard, M. Giorgetta, D. Reinert, J. Klemp, S.-H. Park, W. Skamarock, H. Miura, T. Ohno, R. Yoshida, R. Walko, A. Reinecke, and K. Viner. DCMIP2016: a review of non-hydrostatic dynamical core design and intercomparison of participating models. *Geoscientific Model Development*, 10(12):4477–4509, 2017.

[105] D. Unat, A. Dubey, T. Hoefler, J. Shalf, M. Abraham, M. Bianco, B. L. Chamberlain, R. Cledat, H. C. Edwards, H. Finkel, K. Fuerlinger, F. Hannig, E. Jeannot, A. Kamil, J. Keasler, P. H. J. Kelly, V. Leung, H. Ltaief, N. Maruyama, C. J. Newburn, and M. PericÃ¡s. Trends in data locality abstractions for HPC systems. *IEEE Transactions on Parallel and Distributed Systems*, 28(10):3007–3020, Oct 2017.

[106] F. Váňa, P. Düben, S. Lang, T. Palmer, M. Leutbecher, D. Salmond, and G. Carver. Single precision in weather forecasting models: An evaluation with the IFS. *Monthly Weather Review*, 145(2):495–502, 2017.

[107] M. Wagner, S. Mohr, J. Giménez, and J. Labarta. A structured approach to performance analysis. In C. Niethammer, M. M. Resch, W. E. Nagel, H. Brunst, and H. Mix, editors, *Tools for High Performance Computing 2017*, pages 1–15, Cham, 2019. Springer International Publishing.

[108] N. P. Wedi. Increasing horizontal resolution in numerical weather prediction and climate simulations: illusion or panacea? *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 372(2018):20130289, 2014.

[109] N. P. Wedi, P. Bauer, W. Deconinck, M. Diamantakis, M. Hamrud, C. Kühnlein, S. Malardel, K. Mogensen, G. Mozdzynski, and P. K. Smolarkiewicz. The modelling infrastructure of the Integrated Forecasting System: Recent advances and future challenges. *ECMWF Technical Memorandum*, 760:1–48, 2015.

[110] S. Williams, A. Waterman, and D. Patterson. Roofline: An insightful visual performance model for floating-point programs and multicore architectures. Technical report, Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States), 2009.

[111] J. Wu, P. Wyckoff, and D. Panda. High performance implementation of MPI derived datatype communication over InfiniBand. In *18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.* IEEE, 2004.

[112] W. A. Wulf and S. A. McKee. Hitting the memory wall: Implications of the obvious. *SIGARCH Comput. Archit. News*, 23(1):20–24, Mar. 1995.

[113] C. M. Zarzycki, C. Jablonowski, J. Kent, P. H. Lauritzen, R. Nair, K. A. Reed, P. A. Ullrich, D. M. Hall, M. A. Taylor, D. Dazlich, R. Heikes, C. Konor, D. Randall, X. Chen, L. Harris, M. Giorgetta, D. Reinert, C. Kühnlein, R. Walko, V. Lee, A. Qaddouri, M. Tanguay, H. Miura, T. Ohno, R. Yoshida, S.-H. Park, J. B. Klemp, and W. C. Skamarock. DCMIP2016: the splitting supercell test case. *Geoscientific Model Development*, 12(3):879–892, 2019.

Acknowledgements

The authors would like to thank ECMWF internal reviewers and Alain Joly, Simon Vosper and Piet Termonia for providing very helpful comments that greatly improved the document, Carsten Maass for the Latex templates and Alfred E. Neuman for his ideas and support. Deborah Salmond, Mats Hamrud and George Mozdzynski led the earlier investigations on code design and optimisation.

The Scalability Programme has created a great sense of collaboration on a new and exciting discipline of very high relevance for weather and climate prediction, both at ECMWF and its Member States. The incredible support by young scientists in support of this effort is greatly appreciated.

We acknowledge the support of the European Commission through externally funded projects with the following contract numbers: Future and Emerging Technologies - High-Performance Computing call for research and innovation actions, grant agreements for NextGenIO no. 671951, EuroEXA no. 754337, EPiGRAM-HS no. 801039, ESCAPE no. 671627, ESCAPE-2 no. 800897 and MAESTRO no. 801101; European Research Infrastructures (including e-Infrastructures), grant agreements for ESiWACE no. 675191, ESiWACE-2 no. 823988 and HIDALGO no. 824115; Information and Communication Technologies, grant agreement for LEXIS no. 825532 and CRESTA no. 287703.; ERC grant agreement for Pantarhei no. 320375. We thank the ESCAPE, ESCAPE-2, ESiWACE, ESiWACE-2, EPiGRAM-HS, EuroEXA, HIDALGO, LEXIS, MAESTRO and NextGenIO teams for their collaboration and support.

Some of this research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under contract DE-AC05-00OR22725.

We also gratefully acknowledge access to PizDaint facilitated by Thomas Schulthess and Giuffreda Maria Grazia.

7 Appendix

7.1 Scalability Programme and externally funded projects

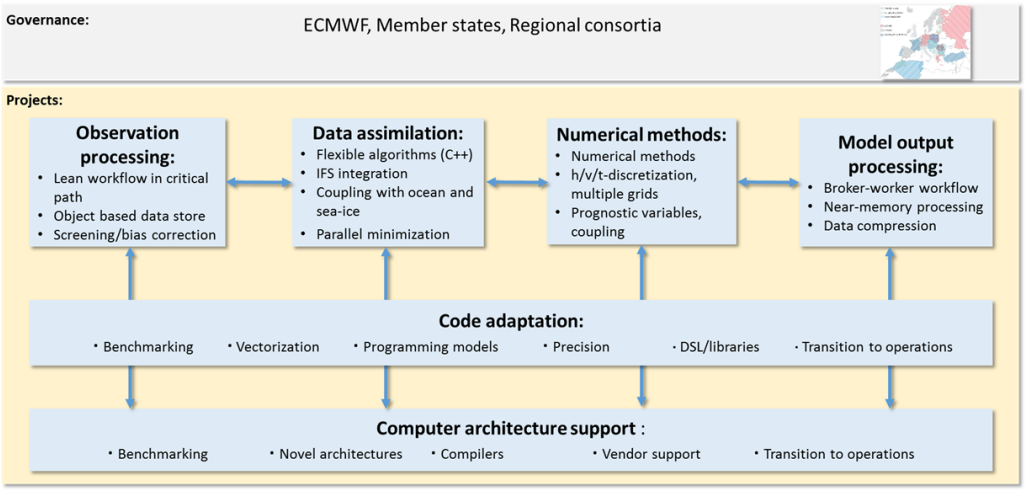


Figure 49: Projects of the first phase of the Scalability Programme, key contents and dependencies.

The Scalability Programme was organised in the following internal projects (Fig. 49):

- The efficient handling of observations prior and during data assimilation is the core objective of the Continuous Observation Processing Environment (COPE) project, in terms of modular and parallelised database access and processing as well as work flow management for both research and operations.
- The Object Oriented Prediction System (OOPS) project addresses the most urgently needed structural code developments supporting future combinations of variational and ensemble methods, but also coupled data assimilation.
- PolyMitos (from Greek *multiple threads*) develops entirely new data structures driven by data locality and the need to avoid communications across large numbers of tasks. It also targets redesigning cost-driving forecast model components applying potentially disruptive changes throughout the dynamical model core and subsequently the whole assimilation and forecasting system.
- The focus of Hermes is to develop a highly efficient and flexible data processing framework, and to provide a scalable way to process datasets with increased size, number and diversity. Hermes prepares capabilities for data processing on heterogeneous architectures and enhanced resilience.
- The Optimisation and Adaptation to Future Supercomputers (OAFS) project represents the linking element between the implementation of new workflows, algorithms as well as code concepts and the evolving hardware architecture. The project includes investment into new programming models to exploit parallelism and vectorisation.
- The Computer Architecture Support (CAS) project manages the interface with heterogeneous architectures that are available inside and outside ECMWF. The project includes code performance evaluation on future processors. CAS is also the main interface to bilateral projects between ECMWF and selected hardware vendors such as Intel, NVIDIA and IBM.

The Programme relied to a large extent on external funding acquired from the European Commission's Horizon-2020 funding programme that complemented the research and development activities described in Section 2 (Fig. 49 and Tab. 4).

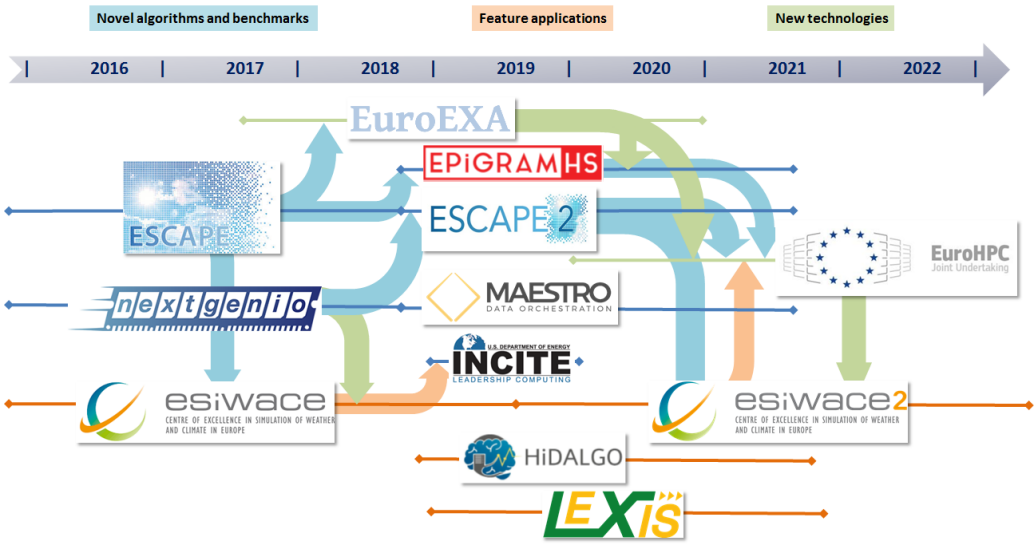


Figure 50: Overview of European Commission funded projects contributing to the first phase of the Scalability Programme, and their inter-dependencies.

Table 4: List of externally funded projects and ECMWF contribution benefit.

Project	Objective	ECMWF contribution/benefit
ESCAPE, ESCAPE-2	Define the next-generation IFS, ICON and NEMO numerical building blocks and compute intensive algorithms, apply compute/energy efficiency diagnostics, identify new approaches and implementation on novel architectures, perform testing in operational configurations, and prepare benchmarks for the EuroHPC HPC infrastructure.	Extract, adapt and optimise present and future IFS dynamical core and physics components on CPU, GPU and optical processors. Assess time and energy to solution performance at scale.
NextGenIO	Define exascale I/O requirements across demanding applications, hardware and data architectures, develop I/O workload simulators, support tools for NVRAM, the necessary systemware and a prototype hardware.	Develop product generation workflow with object based data store, deployable on heterogeneous architectures including NVRAM. Develop and test Kronos I/O workload simulator.
ESiWACE, ESiWACE-2	Provide support, training, services, for fostering community models, tools and software, work towards enhanced code performance at exascale demonstrated by high-resolution simulations.	Assess IFS scalability at very high resolution. Implement and test concurrent execution of physics and wave model, overlapping of computation and data communication using standard programming models, varying numerical precision for data compression.

continued on next page...

...continued from previous page

Project	Objective	ECMWF contribution/benefit
EuroEXA	Co-design a balanced architecture for both compute- and data-intensive applications using a cost-efficient, modular-integration approach enabled by novel inter-die links and the tape-out of a resulting EuroEXA processing unit with integration of FPGA for data-flow acceleration.	Assess portability of IFS components to EuroEXA heterogeneous hardware (incl. FPGA) and evaluate time and energy solution performance and extrapolate to exascale.
EPiGRAM-HS	Validate programming environments for selected applications, extend the programmability and maximise the productivity of application development for large-scale heterogeneous computing systems.	Optimise memory layout, workload distribution and data communication patterns using advanced programming models on heterogeneous architectures for selected IFS components.
MAESTRO	Build a data-aware and memory-aware middleware framework that addresses ubiquitous problems of data movement in complex memory hierarchies and at many levels of the HPC software stack.	Analyse data access and movement patterns in both operational and reanalysis workflows, and integrate them in Kronos simulator. Define data objects and their handling in the workflows by the new middleware.
INCITE	Provide access to the portfolio of US national high-performance computing facilities housing some of the world's most advanced supercomputers.	Assess IFS spectral transform and finite-volume model performance at very high resolution on Summit CPU and GPU.
HiDALGO	Address algorithmic and technological challenges for data-centric-computation, the development and implementation for strong and weak coupling mechanisms, the introduction of AI assisted workflows to improve application lifecycle handling and the integration of real-world sensor data into the simulation execution.	Assess overall simulation - product generation workflow and develop optimisations towards direct streaming of products from model runs, the integration of downstream applications within the prediction workflow, and to provide real-time visualisation of products.
LEXIS	Build an advanced engineering platform at the confluence of HPC, Cloud and Big Data which will leverage large-scale geographically-distributed resources from existing HPC infrastructure, employ Big Data analytics solutions and augment them with Cloud services.	Test ECMWF forecast production workflow in distributed HPC and Cloud computing framework.

The programme also collaborated with the POP and POP-2 centres of excellence hosted by BSC by receiving detailed code performance analyses and optimisation recommendations.

7.2 Programming examples

```
subroutine ec_phys(arg1,arg2,..., argn)
  ...
  call setup_a(arg1,arg2)
  call setup_b(...)
  ...
  call setup_x(...,argn)

  ! All parameterisations in one OpenMP loop

  !$omp parallel
  !$omp do
  do nb=1,num_blocks
    istart = (nb-1)*nproma+1
    iend = nb*nproma
    ...
    call physics_1(istart,iend,blocked_args)
    call update_state(istart,iend,blocked_arg)
    ...
    call physics_2(istart,iend,blocked_args)
    call update_state(istart,iend,blocked_arg)
    ...
  end do
  !$omp end do
  !$omp end parallel
end subroutine ec_phys
```

(a) Current, fully integrated control-flow within a single parallel loop.

```
subroutine ec_phys(arg1,arg2,...,argn)
  ...
  call setup_a(arg1,arg2)
  call setup_b(...)
  ...
  call setup_x(...,argn)

  ! Each parameterisation in a
  ! separate OpenMP loop
  !$omp parallel

  ! First parameterisation
  !$omp do
  do nb=1,arg%num_blocks()
    arg_view = arg%set_view()
    call physics_1(arg_view)
    call update_state(arg_view)
    ...
  end do
  !$omp end do

  ! Second parameterisation
  !$omp do
  do nb=1,arg%num_blocks()
   arg_view = arg%set_view()
   call physics_2(arg_view)
   call update_state(arg_view)
   ...
  end do
  !$omp end do
  !$omp end parallel
end subroutine ec_phys
```

(b) Modular structure with individual parallel loops around current parametrisation kernels.

Figure 51: Pseudo-code for a proposed re-design of the control flow layer of physical parametrisations. A step-by-step migration is proposed that increases modularity of individual sub-sections of the physics code to enable bespoke porting efforts to a single-column format (SCA) and platform-specific parallelization patterns that are generated by a preprocessing infrastructure.

```
subroutine ec_phys(arg1,arg2,...,argn)
  ...
  call setup_a(arg1,arg2)
  call setup_b(...)
  ...
  call setup_x(...,argn)

  ! Each parameterisation is now
  ! written in single-column format

  ! First parameterisation
  !$<sca, mode=cpu, config=<phy1-options>>
  do i=1, arg%ncolumns
   arg_view = arg%set_view()
   call physics_1_sca(arg_view)
   call update_state(arg_view)
  end do
  ...

  ! Second parameterisation
  !$<sca, mode=cpu, config=<phy2-options>>
  do i=1, arg%ncolumns
   arg_view = arg%set_view()
   call physics_2_sca(arg_view)
   call update_state(arg_view)
  end do
  ...
end subroutine ec_phys
```

(a) Modular structure with individual parallel loops around SCA parametrisation kernels.

```
subroutine ec_phys(arg1,arg2,...,argn)
  ...
  call setup_a(arg1,arg2)
  call setup_b(...)
  ...
  call setup_x(...,argn)

  ! Each parameterisation now has
  ! platform-specific annotations

  ! First parameterisation
  !$<sca, mode=cpu, config=<phy1-cpu-options>>
  !$<sca, mode=gpu, config=<phy1-gpu-options>>
  call physics_1_sca(arg1)
  call update_state(arg1)
  ...

  ! Second parameterisation
  !$<sca, mode=cpu, config=<phy2-cpu-options>>
  !$<sca, mode=gpu, config=<phy2-gpu-options>>
  call physics_2_sca(arg2)
  call update_state(arg2)
  ...

end subroutine ec_phys
```

(b) Preprocessor annotations are used to generate platform-specific parallelization patterns.

Figure 52: Fig. 51 cont'd.

```
subroutine physics_1(istart,iend, &
 &                   blocked_args1..n)
  ! first vertical loop
  do k=1,klev
   do i=istart,iend
    little(i,k) = code(i,k)*block(k)
   end do
   do i=istart,iend
    another(i,k) = small(i) * code_block(i,k)
   end do
  end do
  ...
  ! nth vertical loop
  do k=1,klev
   do i=istart,iend
    little = code_block
   end do
   do i=istart,iend
    another = small * code_block
   end do
  end do
end subroutine phys_1
```

(a) Current physics kernel

```
subroutine phys_1_SCA(various,field,arrays)
  ! first vertical loop
  do k=1,klev
   little(k) = code(k)*block(k)
   another(k) = small * code_block(k)
  end do
  ...
  ! nth vertical loop
  do k=1,klev
   little = code_block
   another = small * code_block
  end do
end subroutine phys_1_SCA
```

(b) Single Column Abstraction (SCA) physics kernel

Figure 53: Simplified representations of current and single-column format (SCA) physics parametrisation kernels.

```
integer  :: jnode, jneighbour, inode_neighbour, jlev, nlev
real(wp) :: gradp(:,:), S(:,:,:), vol_inv(:), p(:,:)
real(wp) :: vhat(:,:,:), vstar(:,:,:), pcoeff(:,:,:,:)
real(wp) :: avgQ, Sx, Sy
...
!$omp parallel do schedule( static ) private( jnode, gradp, jneighbour )
!$omp& private( inode_neighbour, Sx, Sy, jlev, avgQ )
do jnode = 1, dstruct%nb_nodes
    gradp(XX,1:nlev) = 0._wp
    gradp(YY,1:nlev) = 0._wp
    do jneighbour = 1,dstruct%nb_neighbours(jnode)
    inode_neighbour = dstruct%neighbours(jneighbour, jnode)
        Sx = S(XX, jneighbour, jnode)
        Sy = S(YY, jneighbour, jnode)
        do jlev = 1,nlev
            avgQ = p(jlev, inode_neighbour)
            gradp(XX,jlev) = gradp(XX,jlev) + Sx * avgQ
            gradp(YY,jlev) = gradp(YY,jlev) + Sy * avgQ
        end do
    end do
    gradp(XX, 1:nlev) = vol_inv(jnode) * gradp(XX,1:nlev)
    gradp(YY, 1:nlev) = vol_inv(jnode) * gradp(YY,1:nlev)

    gradp(ZZ, 2:nlev-1) = (p(3:nlev, jnode) - p(1:nlev-2,jnode)) / (2.0_wp*dz)

    gradp(ZZ, 1:nlev:nlev-1) = (vhat(ZZ,1:nlev:inlev-1,jnode) -                &
        pcoeff(ZZ,XX,1:nlev:nlev-1,jnode) * gradp(XX,1:nlev:nlev-1) -          &
        pcoeff(ZZ,YY,1:nlev:nlev-1,jnode) * gradp(YY,1:nlev:nlev-1)) /         &
        pcoeff(ZZ,ZZ,1:nlev:nlev-1,jnode)

    do jlev = 1,nlev
        vstar(:,jlev,jnode) = vhat(:,jlev,jnode) -                             &
                              matmul(pcoeff(:,:,jlev,jnode),gradp(:,jlev))
    end do
end do
!$omp end parallel do
```

(a) Fortran original version

```
field [nodes, levels]      p, vhat(3), vstar(3), pcoeff(3,3);
field [nodes, neighbours]  S(2);
field [nodes]              vol_inv;
local                      gradp(3);

foreach node in domain {
    foreach level in range(start_level, end_level) {
        gradp(XX) = neighbour_reduce( op::plus, 0, S(XX) * p ) * vol_inv;
        gradp(YY) = neighbour_reduce( op::plus, 0, S(YY) * p ) * vol_inv;

        if ( level in {start_level,end_level} )
            gradp(ZZ) = ( vhat - pcoeff(ZZ,XX) * gradp(XX)
                               - pcoeff(ZZ,YY) * gradp(YY) )
                        / pcoeff(ZZ,ZZ);
        else
            gradp(ZZ) = ( p[level+1] - p[level-1] ) / (2.0*dz);

        vstar = vhat - matmul(pcoeff,gradp);
    }
}
```

(b) DSL version

Figure 54: Example of porting an existing stencil computation to a yet to be designed domain specific language. This code example is taken from elliptic solver computations in IFS-FVM.