Adv. Sci. Res., 20, 1–8, 2023
https://doi.org/10.5194/asr-20-1-2023

Advances in Science & Research
Open Access Proceedings

Maximizing the potential of numerical weather prediction models: lessons learned from combining high-performance computing and cloud computing

Paraskevi Vourlioti[1], **Stylianos Kotsopoulos**[1], **Theano Mamouka**[1], **Apostolos Agrafiotis**[1], **Francisco Javier Nieto**[2], **Carlos Fernández Sánchez**[3], **Cecilia Grela Llerena**[3], and **Sergio García González**[4]

[1]AgroApps, 55133 Thessaloniki, Greece
[2]Atos, Research and Innovation, ATOS Spain SA, 48013 Bilbao, Spain
[3]CESGA, 15705 Santiago de Compostela, Spain
[4]Atos IT, Research and Innovation, 39011 Santander, Spain

Correspondence: Paraskevi Vourlioti (vvourlioti@agroapps.gr)

Received: 24 August 2022 – Revised: 3 February 2023 – Accepted: 13 March 2023 – Published: 20 March 2023

Abstract. To promote cloud and HPC computing, GRAPEVINE* project objectives include using these tools along with open data sources to provide a reusable IT service. In this service a predictive model based on Machine learning (ML) techniques is created with the aim of preventing and controlling grape vine diseases in the wine cultivation sector. Aside from the predictive ML, meteorological forecasts are crucial input to train the ML models and on a second step to be used as input for the operational prediction of grapevine diseases. To this end, the Weather and Research Forecasting model (WRF) has been deployed in CESGA's HPC infrastructure to produce medium-range and sub-seasonal forecasts for the targeted pilot areas (Greece and Spain). The data assimilation component of WRF – WRFDA – has been also introduced for improving the initial conditions of the WRF model by assimilating observations from weather stations and satellite precipitation products (Integrated Multi-satellitE Retrieval for GPM – IMERG). This methodology for assimilation was developed during STARGATE* project, allowing the testing of the methodology in the operational service of GRAPEVINE. The operational production of the forecasts is achieved by the cloudify orchestrator on a Kubernetes cluster. The connections between the Kubernetes cluster and the HPC infrastructure, where the model resides, is achieved with the croupier plugin of cloudify. Blueprints that encapsule the workflows of the meteorological model and its dependencies were created. The instances of the blueprints (deployments) were created automatically to produce operationally weather forecasts and they were made available to the ML models via a THREDDS server. Valuable lessons were learned with regards the automation of the process and the coupling with the HPC in terms of reservations and operational production.

1 Grapevine methodology overview

In the era of big data and cloud computing, a pressing demand has risen for systems and pipelines capable of handling different types of models, data and data formats to produce meaningful outcomes for the user in an operational manner. The importance of this goal is highlighted, amongst others, by the UN Sustainable Development Goals (SDG) agenda and the European Green transition, focusing on Digital Ecosystems (DEs) and particularly Digital Twins that provide flexible and operative frameworks (Nativi et al., 2021). The framework of GRAPEVINE aligns with the concept of Digital Twins (DT) of the earth and the initiative Destination Earth – also known as DestinE. Conceptually, the goal of Destination Earth is to develop a dynamic, interactive, multi-dimensional, and data intensive replica of the Earth (system), which would enable different user groups (public, scientific, private) to interact with vast amounts of natural and socio-economic information. A common infrastructure that provides access to data, sophisticated computing (including high performance computing), software, AI applications, and analytics is at the core of Destination Earth. Additionally, this infrastructure supports numerous DTs that replicate various elements of the Earth system, including weather prediction and climate change, the security of food and water, the global ocean circulation, the biogeochemistry of the oceans, and more yet-to-be-defined elements (Nativi et al., 2021). In GRAPEVINE we followed and demonstrated the interconnected use of cloud and k8s clusters (as was in a prototype of the DT; Nativi et al., 2021) by creating blueprints to develop an operational or on-demand service, which can be executed, combing HCP cloud and AI applications.

Data driven and physic-based models that have different types of requirements in computational power and software packages can work in a harmonized way. This can be achieved by delivering tools that combine the computational power of HPC and the services of cloud computing. For the latter, the cloud provisions tools that connect to the HPC and manage the workloads of the models in the HPC while dictating the sequence of the operations for the targeted output. In the heart of this process lies GRAPEVINE, a project that elegantly demonstrates how physics-based models such like numerical weather prediction models can be operated in HCP centers, by defining blueprints of the processes in a Kubernetes cluster's orchestrator. The produced information is then accessible to data driven models that inform the user for potential risks of grapevine diseases.

In the work of this paper, particular focus is placed on the weather forecasting service of Grapevine, creating a workflow that combines cloud computing with the services of HPC. The weather forecasting production aims to deliver weather forecasts in a timely manner and thus necessitates considerable computational power - a computational power that increases for higher spatio-temporal resolutions. Subsequently, it is considered as a traditional HPC application. Cloud computing represents a paradigm shift in several areas: corporations are adopting Cloud Computing's pay-per-use model instead of constructing in-house datacenters while dynamic, on-demand, and adaptable infrastructures are replacing datacentres (Monteiro et al., 2015). The migration of traditional HPC application such as the weather forecasting to the cloud is not a straightforward procedure. To generate templates, running systems must be virtualized, operating systems must be tuned for the new architecture, and applications must be adapted to cloud standards (Monteiro et al., 2015). However, cloud computing offers developers and users of HPC numerous potential benefits.

These benefits include the dynamic acquisition of computing and storage resources and the access to scalable services. On top of this, cloud services can abstract away the underlying system and automate deployment and control of supported software and service, a process that takes weeks to configure in conventional datacentres (Bunch et al., 2011). In this context, GRAPEVINE has developed a methodology to provide an operational service that uses both cloud and HPC power and it is – to the authors' knowledge – a unique case. The hypothesis of our work is that by facilitating the execution of agriculture-related models in cloud and HPC, we can enable new features and improve the accuracy of such models, becoming more useful for this ecosystem. More specific, the developed methodology highlights the use of HPC, instead of abolishing the HPC with Virtual Machines on the cloud. The development showcased that despite some challenges found in HPC and CPU hours reservations, the cloud can be used to create blueprints of procedures that are executed in an operational manner in HPCs, thus providing a pathway to create services on cloud that involve computational demanding tasks in HPCs. Interestingly, and considering the number of HPCs centers in Europe, this GRAPEVINE service showcased that a setback regarding the use of a particular HPC allowed to seamlessly transfer the service from one HPC center to another with the help of the methodology based on "blueprints" that reside on the Kubernetes cluster.

The Grapevine processing pipeline comprises the HPC infrastructure where the models reside, the Kubernetes cluster that hosts the orchestrator and a platform where the results are showcased (see Fig. 1). The models include both data driven and physically based systems. Phenology models describing the phenology state of grapevines and disease models forecasting the on-set of diseases like botrytis and blackrot amongst others are operationally executed as dictated by the orchestrator (Moshou et al., 2023). The main driver for the prediction of the phenology state and the on-set of diseases are the prevailing weather conditions at the area of interest. For this reason, a numerical weather prediction model is deployed to produce weather forecasts at high spatial resolutions enclosing vineyards at pilot locations of Aragon in Spain and Central Macedonia in Greece. The topography of the selected pilots necessitates forecasts with high spatial resolutions of up to 2 km to properly resolve the terrain and the earth-atmosphere interactions. Considering that the project has been running operationally the Numerical weather prediction since 2021, the partners have agreed to move to a 6 km spatial resolution weather forecasts for 2022 and onwards to reduce computational consumption. In the next sections the HPC and cloud resources and the methodology to deploy the model on HPC and run it from the cloud are described.

2 High performance computing

CESGA has different computing platforms of different architectures to allow the researcher to always choose the architecture that best suits their calculation needs. For simulations that require calculation of high performance and supercomputing, the FinisTerrae (generic name of the different

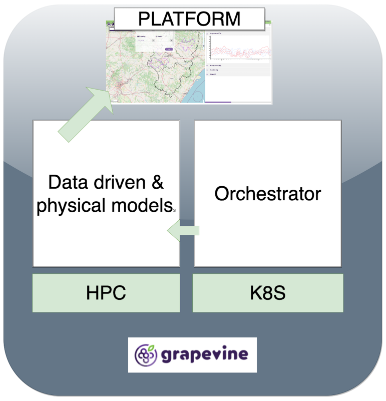

Figure 1. Grapevine high-level architecture.

generations of CESGA's supercomputers) offers higher performance and a high-performance interconnection network for parallel work or work that requires the use of GPUs (Centro de Supercomputation de Galicia, 2022). FinisTerrae also allows for simulations that require the handling of large volumes of data. It is an advanced computer equipment, integrated in the Singular Technical Scientific Facility (ICTS), Spanish Supercomputing Network (RES). It is a Bull ATOS bullx equipment distributed in 8 racks or cabinets and has 320 computer nodes, 7712 cores, 44 544 GB of memory and 750 000 GB of high-performance storage Lustre. All processing and computing nodes are interconnected through a Mellanox Infiniband FDR low latency network. The peak computing capacity of the equipment is 328 272 Gflops and the sustained performance obtained in the Linpack test is 213 000 Gflops.

GRAPEVINE project uses FinisTerrae II (and FTII) HPC system at CESGA to run the weather forecast simulations. The system has the following configuration nodes, the specifications of each of those nodes are:

- Node's Model: R424;
- Node's Processor Model: Intel® Xeon® E5-2680 v3 E5-2680v3 12c;
- number of cores per node: 24;
- RAM per node: 128 GB.

The weather forecasting model, empowered by CESGA's HPC FinisTerrae, is able to produce high resolution weather forecasts of up to 7 d and sub-seasonal forecasts of up to 2.5 months. The number of nodes involved in the jobs run from the weather forecasting model ranges from 4 to 14 nodes depending on the part of the model being executed. The model is executed twice a day, updating the initial conditions with the most recent atmospheric information. To do so, jobs are submitted to the software manager of the HCP, the slurm workload manager (Yoo et al., 2003). The procedure includes adding SBATCH commands to the scripts which inform the slurm manager about the cores needed, the number of nodes, the allocated time on the cores and when needed the reservation name. This is the typical way to submit jobs to the HPC, which in the framework of the GRAPEVINE project was replaced by the orchestrator, described later.

To configure the job submission needs, scaling experiments were conducted, and the optimal combination of time and cores needed was selected. These needs come with close relation to the spatial resolution and geographical extension of the selected domain. The model configuration included the telescoping nesting in which a coarser domain of 18 km was dynamically downscaled to a finer resolution domain of 6 km enclosing the pilot areas of the project. The weather forecasting model was able to deliver with an hourly output of 7 d forecast (raw data) and an allocation of 336 cores for 3.5 h.

To provide an operational framework to the weather forecasting service, the necessity of reservations to the HPC system became apparent. When a job is submitted to the HPC system, it is not executed immediately but it is assigned a priority from the slurm workload manager according to internal rules and based on the availability of nodes/cores in the system. The reservations are done after request to the dedicated HPC department that books the requested number of cores for a specific amount of time, a date, and a starting time. The booking procedure comes with a dedicated name of the reservation, assigned to the user of the HPC. This name is added as an SBATCH command when using the scripts to submit the jobs or through the blueprints (discussed later) when jobs are submitted through the orchestrator.

The reservation task needed the seamless cooperation of the three partners involved: CESGA created a bi-weekly list of daily reservations as designed by AgroApps and were automatically updated in the instances of blueprints of the orchestrator by ATOS. This requires close monitoring by the involved partners and is an additional process that needs consideration when planning operational systems with involvement of HPC centres. Another point of caution would be which jobs will be run under reservation and which not. The heavy jobs that need a lot of resources are definitely submitted with reservations while light jobs are sent in the system's queue. For example, some input observational data that can be downloaded via ftp do not need to have a reservation as they are not computationally heavy jobs. This configuration spares booking for a long time the HPC but requires extra blueprints to be submitted. As an example, the weather forecasting model requires input from global forecasting models (see Sect. 4) and these data are available daily early in the morning. For this process we submit a blueprint with the sole purpose to download the data with no reservation. Later in the morning some necessary observational data are made available, and this is when we submit a second blueprint with a reservation to execute the model.

As will be discussed in the Weather Forecasting Architecture Section, the weather forecasting modelling framework has a modularity, meaning dedicated scripts carry out sub-tasks. Each sub-task may depend on a previous job to finish and takes a different number of resources from the HPC to complete the job. When designing the submission process and the reservations in such a modular framework, a trade-off must be considered between the time of the HPC allocation and number of reservations. For example, the pre-processing part of the forecasting model takes 24 cores for fifteen minutes and when this job finishes, the heavy part with the 336 cores that performs the weather forecasting, starts. The question in this situation is how to handle reservations when many such small jobs precede the heavy job. One way would be to create a blueprint for each sub-task and a dedicated reservation name. This solution raises three problems: firstly, it adds the complexity of creating multiple reservation names; secondly, creating several reservations at the same time does not ensure that they are done on the same node; and thirdly, the utility of the orchestrator to define dependencies within one blueprint with many jobs is lost. The latter translates to submitting multiple instances of the blueprints from the orchestrator and makes monitoring of the processes harder. Another way to deal with this is to create one reservation with one name and occupy the maximum number of cores the heaviest job needs and add all smaller jobs to the same reservation. Of course, this means that, while the smaller jobs are executed, a great number of reserved cores will remain idle and unreachable to other HPC users. If the time the smaller jobs take is small enough this solution is acceptable, but a limit should be placed in order not to waste valuable core-hours from the HPC community.

Another question that had to be addressed was the accounting of HPC hours inside and outside reservations:

– if no reservation is made the account of HPC hours is similar to this formula:

$$\begin{aligned}\text{HPC hours} =& \text{duration of the job's run} \cdot \text{number} \\ & \text{of nodes} \cdot \text{number of cores per node;} \end{aligned} \tag{1}$$

– if the user does not delete the reservation once the job ends the account of HPC hours is similar to this formula:

$$\begin{aligned}\text{HPC hours} =& \text{duration requested} \cdot \text{number of nodes} \\ & \text{requested} \cdot \text{number of cores per node} \\ & \text{requested;} \end{aligned} \tag{2}$$

– if the user deletes the reservation once the job ends the account is similar to this formula:

$$\begin{aligned}\text{HPC hours} =& \text{time since the reservation starts to the} \\ & \text{deletion of the reservation} \cdot \text{number} \\ & \text{of nodes requested} \cdot \text{number of cores} \\ & \text{per node requested.} \end{aligned} \tag{3}$$

This means that the user must delete the reservation once the jobs end or the full amount of resources requested will be accounted for, even if they are not used, as the resources will not be available to other users. It is one of the tasks of the Orchestrator (or the user) to delete the reservation once the jobs end so as not to waste resources. In the following section a description of the Kubernetes cluster and the components utilised to communicate with the HPC, where the models and scripts reside, is detailed.

3 Orchestrator in Kubernetes cluster

The orchestrator is a key component in the GRAPEVINE infrastructure that is responsible for the execution of computational workflows on the available resources, namely Cloud VM resources provided by the GRAPEVINE K8S cluster and HPC resources provided by CESGA's HPC system FinisTerrae II. After CESGA's migration to FinisTeraeIII, Grapevine tool was successfully re-deployed also in the new infrastructure. The Kubernetes (K8s) cluster was deployed on CESGA premises to run the Orchestrator and consists of 8 virtual machines dedicated to the cluster configured as follows:

– 3 master nodes with 4 VCPU, 8 GB RAM, 80 GB of disk;

– 5 worker nodes with 8 VCPU, 16 GB RAM, 80 GB of disk.

The orchestrator is crucial for the execution of all the models in the project on these resources. In that sense, the orchestration optimises the execution of the workflows by modularizing them and dividing them into sizeable chunks that can be run as jobs on back-end nodes on CESGA. The added value of using the orchestrator is that the dependencies between jobs can be easily specified, such that different parts of the workflow can be executed as jobs running in parallel or sequentially. The orchestrator is a specific plugin for Cloudify (Cloudify, 2022) that enables the usage of HPC, supporting scheduled execution of the workflows at a specific time, such that the GRAPEVINE infrastructure will be able to always show results generated on the latest available raw data. Additionally, thanks to its connectors for Slurm, PBS Pro and Torque, it provides an abstraction layer that makes easier to change the location to execute the jobs (abstracting the complexity of HPC systems to the user). Everything is secured by using Keycloak for users management and Vault for users' credentials storage (so the orchestrator can take credentials directly, not requiring users to provide them for each execution). On top of these features, the orchestrator supports data management as it includes a built-in data mover component based on gridFTP (and to access data from ECMWF, if required). This component allows for secure and fast data transfer within different infrastructure components deployed

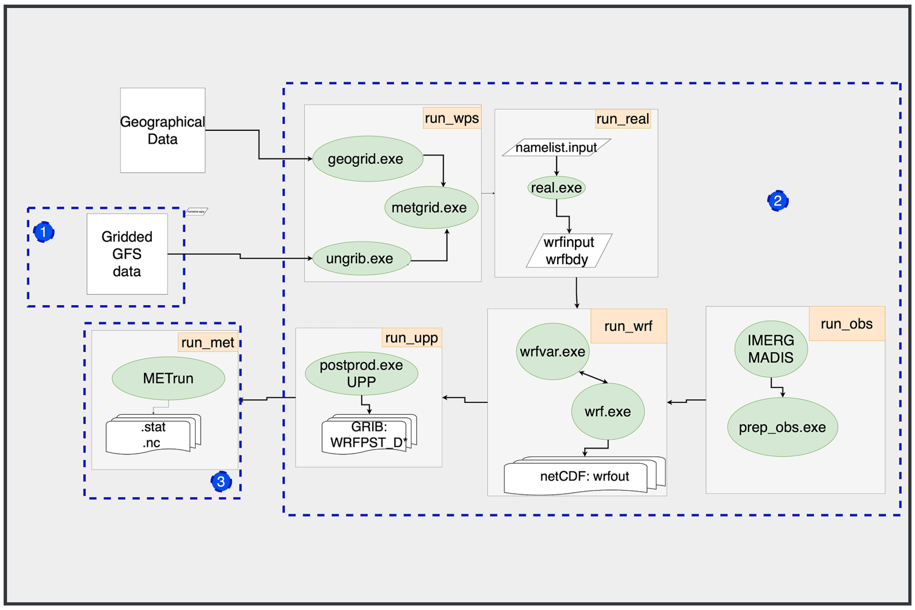

Figure 2. Weather forecasting components (in green). Group of components that belong to the same sub-process are under the same grey box and the sub-process can be seen as run_[name_of_subproces]. With punctuated blue lines, numbered 1, 2 and 3, are the blueprints for the orchestrator. Each blueprint encloses a number of components of the model.

in CESGA, such as the shared hard drive, where the computational results are stored, and the THREDDS Web Data Server (Unidata THREDDS Data Server, 2022), where the results are regularly transferred so they can be queried by the Front-end/Visor of the project. Moreover, the Orchestrator has been integrated with the monitoring component based on Prometheus (Prometheus, 2022). We have implemented a new exporter for the Prometheus collector that is able to retrieve metrics from different schedulers (mainly, PBS Pro and Slurm). Such exporter makes use of different features of the schedulers commands in order to collect information about the jobs and the queues (job execution time, job queue time, job exit status, number of CPUs used by the job, nodes available in the partitions, status of the queues, etc.). The Orchestrator can also send metrics to Prometheus through its pushgateway interface, if required (e.g. when starting and finishing jobs).

The Orchestrator has been following an integration process with respect to two main elements: the meteorological model and the Data Scientific ML Engine. In the case of the meteorological model, such a model is seen as an application to run through the Orchestrator. Therefore, first the pipeline of the meteorological model was split into modular scripts and afterwards blueprints were created that define the relationship between the tasks to be executed on the HPC. The blueprints were created based on the TOSCA language, to which some specific extensions were defined, enabling the usage of HPC. Such extensions are totally in line with the standard, defining a few new types of tasks (for jobs, data sources and data transfer protocols) and exploiting the relationships that can be defined for the tasks (so we may force input data movement before running a job, for instance). In Fig. 2, the components of the weather forecasting framework are grouped to make sub-processes that belong to the same blueprint.

This workflow, then, is installed in the Orchestrator and instantiated by providing the required inputs. The communication with the HPC is achieved via the Croupier plugin of Cloudify for HPC and batch job orchestration (Git Hub croupier, 2022). The orchestrator interface allows an easy monitoring process for the admin of the Grapevine tool as it provides a web access where all the jobs and their status are displayed. Such interface is complemented with Grafana dashboards connected to the Prometheus, with information about the infrastructure and the jobs. This feature allows the admin to surpass the connection to the HPC infrastructure (requiring VPN access) and simply submit again a new job from the web interface in case a job fails. This applies to blueprints that do not have a reservation name defined. Meaning that in case of failure, a set of identical to the original blueprints should be set but without reservation information.

4 Weather forecasting architecture

The architecture of the weather forecasting system is illustrated in Fig. 3. The core system is based on the Weather and Research Forecasting system (WRF) (Skamarock et al., 2019). WRF is designed for both operational and atmospheric research applications. WRF modelling framework supports two dynamical cores, a data assimilation system,

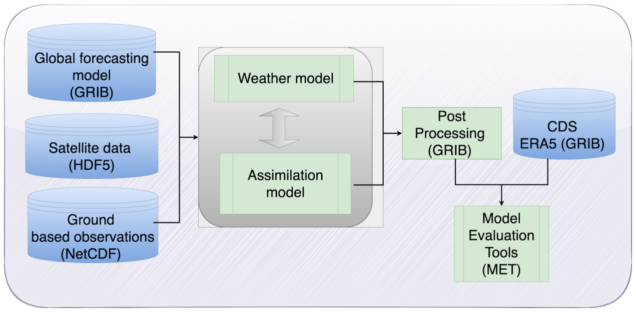
Global forecasting model (GRIB)
Satellite data (HDF5)
Ground based observations (NetCDF)
Weather model
Assimilation model
Post Processing (GRIB)
CDS ERA5 (GRIB)
Model Evaluation Tools (MET)

Figure 3. Weather forecasting model architectural design.

and a software architecture supporting parallel computation and system extensibility. Both the weather forecasting model WRF and its assimilation component (WRFDA) have been deployed in CESGA's HPC FinisTerrae. The assimilation model, as can be seen in Fig. 3. is fed with observational data and creates the initial conditions for the forecasting model by considering both observations and the background model state. The WRF dynamical core is then fed along with the new initial conditions the boundary conditions from the Global Forecasting System (GFS) (NCEI, 2022) and integrates in time the atmospheric state, producing the medium range forecasts. The hourly output is then post-processed in a common meteorological format (GRIB format). At the final stage, meteorological fields (temperature, wind speed, relative humidity, and precipitation) as forecasted by the model are compared with the reanalysis data ERA5 (Hersbach et al., 2020) from the Copernicus Data Store (CDS) (CDS, 2022) and scores like Mean Absolute Error and Root Mean Square error are produced.

The deployment of the framework is based on bash and python scripts, each responsible for a sub-task. The scripts reside in dedicated paths in the HPC and are executed by the orchestrator. The scripts can be split into three categories:

1. download data scripts;
2. data pre-processing scripts (for observations to be fed into the models);
3. model execution scripts.

Satellite, weather station data and GFS data are downloaded via FTP with simple bash scripts and erased after usage. The reanalysis data are downloaded via the cds api and dedicated python scripts to download the data. The pre-processing scripts include fortran compiled code to read netCDF and HFD5 data and convert them to a suitable WRFDA format (little-r). For weather stations data conversion, the madis-to-little-r utility (WRFDA Users Page, 2022) is applied. For the satellite data conversion, a dedicated fortran compiled code was created that cuts the information to the domain of interest and converts the rain rate to 6 h accumulation and writes the output into the little-r format. The model execution scripts include the data handling, moving appropriate executable files from residing paths in HPC to the working folders of the HPC system. Once the submitted jobs are finished, the produced data are moved to a shared folder to all partners allowing them to use these data for training and forecasting purposes. The complexity of the meteorological output format created the need for an additional step, the THREDDS server instance. The THREDDS server provides catalogue, metadata, and data access services for scientific data, allowing thus an easy access to partners needing meteorological information for their models.

The described workflow has in its core compiled code that has performance relationships to the selected compiler (for this work intel compiler) and selected physics packages. To support portability, containerized environments are the optimal solution. HPC systems do support containerization technologies like Docker and Singularity but often these come with some limitations. These include restrictions of building your own image due to permission issues on the HPC and compiler compatibility issues between singularity sand boxes and HPC installed compilers. For the Grapevine project a Singularity image of the verification model was set up by the application department of CESGA. The availability of an official image of the verification model in Docker Hub (Docker Hub, 2022) allowed the easy transfer to a singularity sandbox effortlessly called via the module system of the HPC (module load configurations). For the meteorological model, no official image exists as there are many user-related options to the compilation configuration. This added to the complexity of creating first the image on the HPC and then translating it as a singularity sandbox. When one already has set the optimal configuration for the model (at compilation stage), it is best practice to ensure portability by creating singularity recipes or translating Docker images to Singularity sand boxes. Caution and close work with the HPC application department will assure that the compiler issues (the sandbox must have the same compiler type and version as the ones from the HPC system) will pose no issue at execution time.

5 Conclusions and outlook

To fulfil GRAPEVINE's objectives, the involved partners have been closely working to combine research (Numerical Weather prediction and ML for agricultural sector), cloud infrastructure (K8s cluster) and high-performance computing. Numerical weather prediction is a traditional HPC application that demands large amounts of computational resources. In the context of an operational service, that includes physics and data driven models, cloud and HPC can be fruitfully combined as showcased in GRAPEVINE project. Although one could use cloud computing to run a weather prediction model, depending on the resolution and time to execute the model, one would need to handle and set-up large number of resources on the cloud.

HPC centres offer these computational resources and libraries to easily perform such tasks. Weather forecasting can be added to an operational or on-demand service, demonstrated for the first time to our knowledge in Grapevine, by utilising both HCP and the cloud. Cloud resources are acting as the brain of the service, that dictates what is to be run and when taking into account the dependencies of the involved jobs. Such a service though comes with limitations, including the need to create computing reservations into the HCP system, which has challenges and is a possible area of work into automating the processes.

An area of interest lies also in portability of the service. Although the cloud easily connects to different HPC centres, the code has to be present on those clusters to run the model. The service could be enhanced by working with singularity containers (a containerised application for HPC that does not come with the limitations of docker containers regarding root access), allowing easy transfer of the weather forecasting from one HPC centre to another. Simultaneously the same blueprints residing on the cloud could be used to run the weather service to different HPC centres (EGI, 2022).

Finally, the framework of GRAPEVINE aligns with the concept of Digital Twins (DT) of the earth and the initiative Destination Earth – also known as DestinE. In GRAPEVINE we followed and demonstrated how using the cloud and the k8s clusters (as was in a prototype of the DT; Nativi et al., 2021) and by creating blueprints an operational or on-demand service can be executed, combing HCP, cloud, and AI applications. The processes defined in GRAPEVINE have facilitated a straightforward data management and generation that can help to improve the involved models by creating operational or on demand crucial data to train models. Subsequently we believe the lessons we learned from developing such framework are of importance to be sheared with the scientific community.

Data availability. The blueprints can be found in the project's git lab repository: https://gitlab.com/grapevine-project/a3_climate_model_workflow (Grapevine's GitLab repository, 2023). The produced meteorological data, available through THREDDS server can be found here: http://193.144.42.171:8080/thredds/catalog/agroapps_folder/catalog.html (Grapevine's THREDDS server, 2023). Medium range folder data format: YYYY-MM-DD, under folders named UPP and Sub-Seasonal data folder format: S_YYYY_MM_DD.

Author contributions. PV deployed the meteorological model on the HPC, structured the paper, wrote the Weather Forecasting Architecture, parts of all sections and reviewed the paper. TM orchestrated the job execution, co-deployed parts of the model and reviewed the paper. SK overviewed the planning and implementation of the model and reviewed the paper, AA reviewed the paper, FJN and SGG deployed the Orchestrator, configured the blueprints to connect properly with the HPC and wrote the Orchestrator in Kubernetes Cluster section of this paper. Finally, CFS and CGL created the Kubernetes cluster, they deployed the THREDDS server, they created the shared drive, they configured the reservations and wrote the section High Performance computing.

Competing interests. The contact author has declared that none of the authors has any competing interests.

Disclaimer. Publisher's note: Copernicus Publications remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Special issue statement. This article is part of the special issue "EMS Annual Meeting: European Conference for Applied Meteorology and Climatology 2022". It is a result of the EMS Annual Meeting: European Conference for Applied Meteorology and Climatology 2022, Bonn, Germany, 4–9 September 2022.

Acknowledgements. The authors would like to acknowledge the European Union's Horizon 2020 and STARGATE project (no. 818187) for the assimilation methodology developed under the project's scope as well as the Health and Digital Executive Agency (HaDEA) for the funding of GRAPEVINE project (2018-EU-IA-0091) during which the assimilation methodology was tested.

Review statement. This paper was edited by Josef Eitzinger and reviewed by two anonymous referees.

Financial support. The work has been funded by the European Union's Horizon 2020 under reSilienT fARminG by Adaptive microclimaTe managEment (STARGATE) project (grant no. 818187).

References

Bunch, C., Chohan, N., Krintz, C., and Shams, K.: Neptune: a domain specific language for deploying hpc software on cloud platforms, in: Proceedings of the 2nd International Workshop on Scientific Cloud Computing, 8 June 2011, San Jose, California, USA, https://doi.org/10.1145/1996109.1996120, 2011.

CDS – Copernicus Climate Change Service Climate Data Store: ERA5 hourly data on single levels from 1979 to present, https://cds.climate.copernicus.eu/cdsapp#!/home, last access: 25 August 2022.

Centro de Supercomputation de Galicia: CESGA's new FinisTerrae III, https://www.cesga.es/en/cesga-actualiza-el-finisterrae/, last access: 25 August 2022.

Cloudify: Bridging the Gap Between Applications & Cloud Environments, https://cloudify.co/, last access: 25 August 2022.

Docker Hub: Model Evaluation Tools Plus (METplus) Repository, https://hub.docker.com/r/dtcenter/metplus, last access: 25 August 2022.

EGI – European Grid Infrastructure: Advanced computing for research, https://www.egi.eu/, last access: 25 August 2022.

Git Hub croupier: Cloudify plugin for HPCs and batch applications, https://github.com/ari-apc-lab/croupier, last access: 25 August 2022.

Grapevine's GitLab repository: A3_climate_model_workflow, https://gitlab.com/grapevine-project/a3_climate_model_workflow, last access: 17 March 2023.

Grapevine's THREDDS server: Thredds catalog, Grapevine's THREDDS server [data set], http://193.144.42.171:8080/thredds/catalog/agroapps_folder/catalog.html, last access: 17 March 2023.

Hersbach, H., Bell, B., Berrisford, P., Hirahara, S., Horányi, A., Muñoz-Sabater, J., and Thépaut, J. N.: The ERA5 global reanalysis, Q. J. Roy. Meteorol. Soc., 146,, 1999–2049, 2020.

Monteiro, A., Teixeira, C., and Pinto, J. S.: HPC in weather forecast: moving to the cloud, Int. J. Cloud Appl. Comput., 5, 14–31, 2015.

Moshou, D., Paraskevas, C., Kechagias, K., Pantazi, X.-E., Tamouridou, A.-A., Stavridou, D., Pliatsidis, K., Pantazi, E.-G., Fragos, V., Balduque, J., Vourlioti, P., Kotsopoulos, S., and Lacueva-Pérez, F. J.: D3.2 GRAPEVINE Plagues and Diseases Models v2, Zenodo [code], https://doi.org/10.5281/zenodo.7575773, 2023.

Nativi, S., Mazzetti, P., and Craglia, M.: Digital Ecosystems for Developing Digital Twins of the Earth: The Destination Earth Case, Remote Sens., 13, 2119, https://doi.org/10.3390/rs13112119, 2021.

NCEI: Global Forecast System (GFS), https://www.ncei.noaa.gov/products/weather-climate-models/global-forecast, last access: 25 August 2022.

Prometheus: From metrics to insight, https://prometheus.io/, last access: 25 August 2022.

Skamarock, W. C., Klemp, J. B., Dudhia, J., Gill, D. O., Liu, Z., Berner, J., Wang, W., Powers, J. G., Duda, M. G., Barker, D. M., and Huang, X.-Y.: A Description of the Advanced Research WRF Version 4, NCAR Tech. Note NCAR/TN-556+STR, NCAR, 145 pp., https://doi.org/10.5065/1dfh-6p97, 2019.

Unidata THREDDS Data Server: THREDDS Data Server (TDS), https://www.unidata.ucar.edu/software/tds/current/, last access: 25 August 2022.

WRFDA Users Page: Using MADIS observations with WRFDA and/or WRF, https://www2.mmm.ucar.edu/wrf/users/wrfda/download/madis.html#updates, last access: 25 August 2022.

Yoo, A. B., Jette, M. A., and Grondona, M.: Slurm: Simple linux utility for resource management, in: Workshop on job scheduling strategies for parallel processing, Springer, Berlin, Heidelberg, https://doi.org/10.1007/10968987_3, 2003.