AGINFRA PLUS: Running Crop Simulations on the D4Science Distributed e-Infrastructure

M. Knapen, Rob M. Lokers, Leonardo Candela, Sander Janssen

AGINFRA PLUS: Running Crop Simulations on the D4Science distributed e-Infrastructure

Rob Knapen[1], Rob Lokers[1], Leonardo Candela[2], and Sander Janssen[1]

[1] Wageningen University and Research, Wageningen, The Netherlands
[2] ISTI – National Research Council of Italy, Pisa, Italy

Abstract.
Virtual Research Environments (VREs) bridge the gap between the compute and storage infrastructure becoming available as the 'cloud', and the needs of researchers for tools supporting open science and analytics on ever larger datasets. In the AGINFRA PLUS project such a VRE, based on the D4Science platform, was examined to improve and test its capabilities for running large numbers of crop simulations at field level, based on the WOFOST-WISS model and Dutch input datasets from the AgroDataCube. Using the gCube DataMiner component of the VRE, and based on the Web Processing Service standard, a system has been implemented that can run such workloads successfully on an available cluster, and with good performance, providing summarized results to agronomists for further analysis. The methods used and the resulting implementation are briefly described in this paper. Overall the approach seems viable and opening the door to many follow-up implementation opportunities and further research. Some of them are indicated in more detail in the conclusions.

Keywords: Distributed Computing, e-Infrastructure, Virtual Research Environment, Crop Simulation Model, WOFOST.

1 Introduction

In [10], it is argued that in Agronomy the major Big Data challenges are with variety and veracity, and that tackling the issues with volume and velocity of the data is more generic and to be solved with common industrial Information Technology solutions. Yet, such solutions still have to be created, adapted to, and tried in the field of Agro Informatics. Besides, agronomists and researchers of related domains have to be introduced to them and start making use of these new technologies.

Meanwhile the range of options is getting broader with the rise of container technologies such as Docker (https://www.docker.com), Kubernetes (https://kubernetes.io), and Singularity (https://sylabs.io), and Cloud computing. Platforms familiar to computer scientists and software engineers such as Google Cloud Platform (https://cloud.google.com), Amazon Web Services (https://aws.amazon.com), Microsoft Azure (https://azure.microsoft.com/), are extending their reach by adding tools interesting to other researchers, targeting various domains. While other vendors make

more basic compute and storage resources still easier accessible at low prices. However, the high-level platforms have the risk of vendor lock-in and unknowingly handing over of data, while the low-level solutions (including direct use of High Performance Computing) require IT proficiency to make proper use of them.

The European Open Science Cloud (EOSC) is an initiative that addresses such issues and promotes a more open, federated, research infrastructure that gives access to compute and storage resources, and support the ideas of Open Science and FAIR data sharing (https://www.force11.org/group/fairgroup/fairprinciples). It is in line with those thoughts that the D4Science platform [3] is being developed.

In the European H2020 AGINFRA PLUS project [2] the D4Science technology is used to create a number of Virtual Research Environments (VRE) specifically targeting research communities in the food and agriculture domain, with the goal to accelerate user-driven innovations of the existing e-Infrastructure. A number of key use cases were selected based on typical work and requirements within each community. This paper will further focus on one of those use cases, being the ability to run crop simulation models at scale on a compute cluster hosted by D4Science, allowing for horizontal scalability to handle large workloads such as running crop simulations for all the crop parcels in the Netherlands.

2 Methods

2.1 D4Science Cloud Computing e-Infrastructure

D4Science [5] promotes open science through the operation of an innovative data and compute infrastructure service, build using the gCube framework [8]. The gCube technology allows easy construction and development of VREs. A VRE in general is a web-based working environment tailored to support the needs of their designated communities, each working on specific research questions. The VRE offers users with domain-specific facilities, e.g. certain computational algorithms, and typically needed datasets, integrated with more common services that support collaboration and cooperation amongst users, e.g. a shared Workspace to store and organize versions of research artifacts, a social networking area, a data analytics platform, and a catalogue-based publishing platform (see [3]).

The data analytics platform (gCube DataMiner, see **Fig. 1**) is of special interest, since this is the service used to run the crop simulation model. There are two ways to interact with it, one is at a low level where algorithms have to be written using specific Java interfaces that are set up to do map-reduce types of processing [7]. Such algorithms gain access to the 'Worker' cluster, a set of thin nodes ('slow' CPU and limited memory) specifically targeted at this type of data processing. The other option is to write algorithms that can be deployed using the DataMiner importer, which wraps it and deploys it as a Web Processing Service (WPS, a standard from the Open Geospatial Consortium (OGC)) on the 'Master' cluster. These are fat nodes (fast CPU and lots of memory) in the system that handle the queuing, load balancing, and execution of WPS requests.

For both approaches DataMiner automatically provides recording of provenance information using the W3C Prov-O standard [11], so that any run of an algorithm can easily be repeated.

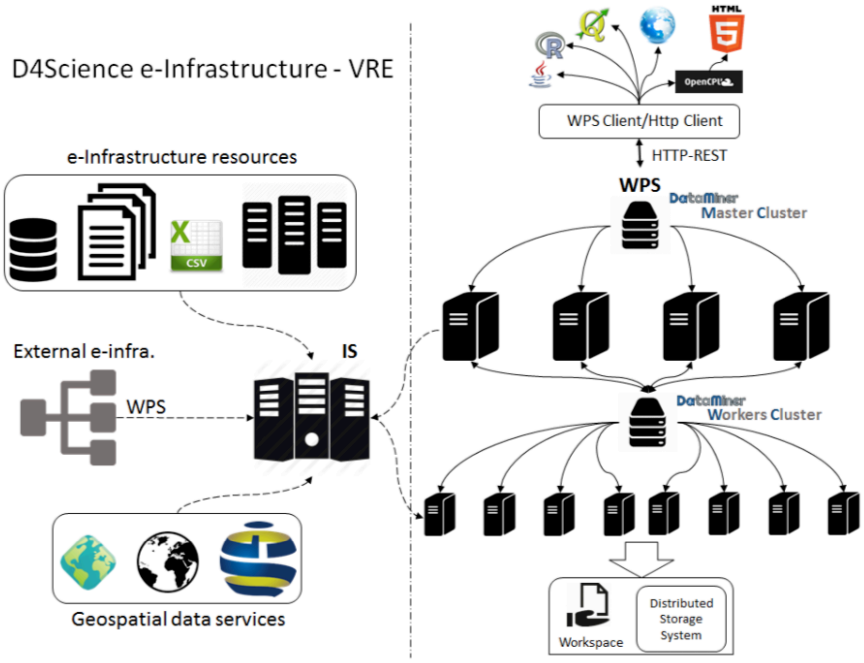

Fig. 1. Architecture of the gCube DataMiner system. With on the right side the two compute clusters and the distributed storage system, and on the left side several e-Infrastructure resources which are indexed on the D4Science Information System (IS).

2.2 The Web Processing Service standard

The Web Processing Service interface standard is one of the standards of the Open Geospatial Consortium. It is lightweight and XML-based, and easy to use to publish, discover, and execute processes as a service. Processes (e.g. calculations, algorithms, simulation models) can be both simple and complex. Input data for the processes can be included in the HTTP POST request to the WPS service or be made accessible via HTTP URLs. WPS services can also be integrated into workflows by workflow management systems such as Galaxy (galaxyproject.org), Knime (www.knime.com), and Apache Taverna (taverna.apache.org).

WPS has been designed to work with spatially referenced data, but it can be used with any kind of data. It also was not designed to work with distributed computing

systems. The main operations defined by the interface are: **GetCapabilities**, **DescribeProcess**, and **Execute**. Allowing to get the list of available services on a system, get a description of the required inputs and the produced outputs, and to execute the process. These do not allow for very dynamic use of a compute cluster. E.g. all outputs of a process have to be specified at process description time, and it is hard to track how busy the system is and how many new requests it would be able to process quickly. In D4Science some of these issues have been addressed.

2.3 WOFOST-WISS

WOFOST-WISS is a new implementation of the well-known and widely used WOFOST crop growth model [6]. It is built on top of the Wageningen Integrated Systems Simulator (WISS), a Java-based, lightweight simulation model framework targeting the agro-economical modelling domain. WOFOST-WISS ensures high numerical performance and robustness, both required for large scale operational application of crop models. It is a mechanistic, dynamic model that explains daily crop growth on the basis of the underlying processes, such as photosynthesis, respiration, and how these processes are influenced by environmental conditions.

Since it is written in Java (openjdk.java.net), WOFOST-WISS needs a Java Runtime Environment (JRE) to run it. This makes it easy portable to any computing platform for which a JRE is available (almost all). The code is executed by a Java Virtual Machine (JVM), which allows for runtime code optimizations, as well as remote debugging, and real-time performance monitoring. It also makes the model easy to integrate into the Java eco-systems and use it from other JVM-based programming languages.

An essential design principle of WOFOST-WISS is that the model components are stateless, and all state is securely kept into an object called **SimXChange.** All input data is provided to the model in a **ParXChange** object. The model itself has no side-effects, hence, from a Function Programming (FP) perspective it can be regarded as a pure function. This makes it well suited for use in distributed computing.

2.4 AgroDataCube

Given that for many applications in the agri-food domain the same basic, large, datasets are always needed, at Wageningen Environmental Research these are being made available as harmonized open data in the AgroDataCube [9], including information about crop fields, crop growth (as indexes derived from remote sensing data), observed weather, terrain height, and soil conditions. Using an access token all data can be retrieved by HTTP GET and POST requests in GeoJSON format (geojson.org) from a REST (https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm) based Application Programming Interface (API). It is also possible to retrieve sub-field gridded data such as the detailed 10m x 10m Normalized Difference Vegetation Index (NDVI).

For the Netherlands the AgroDataCube contains in principle all the data for multiple years needed to run crop simulations at the crop field level. It does however require some further pre-processing of the data, including the mapping of crop codes into the

needed crop specific parameters, completing available weather data into a full daily time series of all required weather variables, and calculation of essential soil characteristics from the available soil information.

2.5 Functional Programming and the Actor Framework

The technologies available for building applications continue to evolve at a rapid pace. Systems such as D4Science make it possible to effectively utilize clusters of cores on individual servers and clusters of servers that work together as a single application platform. Costs for memory and disk storage have dropped, and network speeds have grown significantly. This allows for large volumes of data to be collected and processed. However, timely processing typically cannot be done any more on a single traditional computer with limited (vertical) scalability. It requires distributed computing on multiple computers that allow better (horizontal) scalability. And distributed computing is difficult to program without using new programming paradigms such as functional programming.

Functional Programming (FP) is a style of writing computer programs that treats computation as the evaluation of mathematical functions and avoids changing state and mutable data, and side-effects in general. Programming is done by writing expressions or declarations, instead of statements. FP stems from lambda calculus, a formal system developed in the 1930s to investigate computability, amongst others. Lisp (lisp-lang.org), Clojure (clojure.org), Erlang (erlang.org), Haskell (haskell.org), and Scala (scala-lang.org) are some well-known functional programming languages. The latter is a JVM based programming language, hence offers good integration with Java.

Because it avoids side-effects FP fits well for writing programs that are run in distributed computing environments. It is much easier to reason about the execution of (pure) functions, than about multiple computers and threads processing the objects of a program written in an Object-Oriented programming language. Combining FP with a message passing architecture such as the Akka framework [1] allows even further abstraction and isolation, away from the low-level wiring and very technical programming techniques otherwise needed to make efficient use of computational resources.

3 Results

As described, one of the use cases in the AGINFRA PLUS project was to pilot running crop simulations for the crop fields of the Netherlands, using the D4Science distributed infrastructure. Within the project a small cluster was available for testing purposes, with 6 fat nodes, and a slightly larger cluster for operational deployments. A number of VREs have been created, among which one for the Agro-Environmental modelling use cases. This VRE contained the generic components such as the social messaging, the shared Workspace, and DataMiner, as well as more domain specific tools such as access to SoilGrids (soilgrids.org), Jupyter notebooks (jupyter.org), and customized visualizations.

It would certainly have been possible to use the map-reduce approach of DataMiner for running many crop simulations on the system. However, for the long term, this appeared to be a less flexible solution, binding the software directly to the D4Science platform. Being a research organization, future projects might involve running crop simulations in other environments than D4Science, and not in a strictly map-reduce way, making it more appealing to use the WPS based processing of DataMiner.

DataMiner runs the WPS processes on available fat nodes in the cluster. For all incoming WPS requests of the same process it handles the queuing and load balancing. Running the crop simulation models efficiently therefore required implementing two types of algorithms for DataMiner. One that can run as many crop simulations simultaneously per fat node once it gets assigned to one, and another type of algorithm that can divide a total workload of crop simulations to be done for a set of selected crop fields, e.g. all maize fields in the Dutch province Limburg (approx. 15.000 for a single year), into smaller batches, start the crop simulation jobs on DataMiner for each batch, and collect and process all results. In essence the two types of algorithms perform the map-reduce processing via the WPS interfaces.

Both algorithms are implemented using the Akka actor framework, so that they make the best possible use of the machines they get assigned to (within limitations set by the system). The 'worker' algorithm (see **Fig. 2**) contains WOFOST-WISS and accesses the AgroDataCube in an optimized way, running as many crop simulations in parallel as possible.

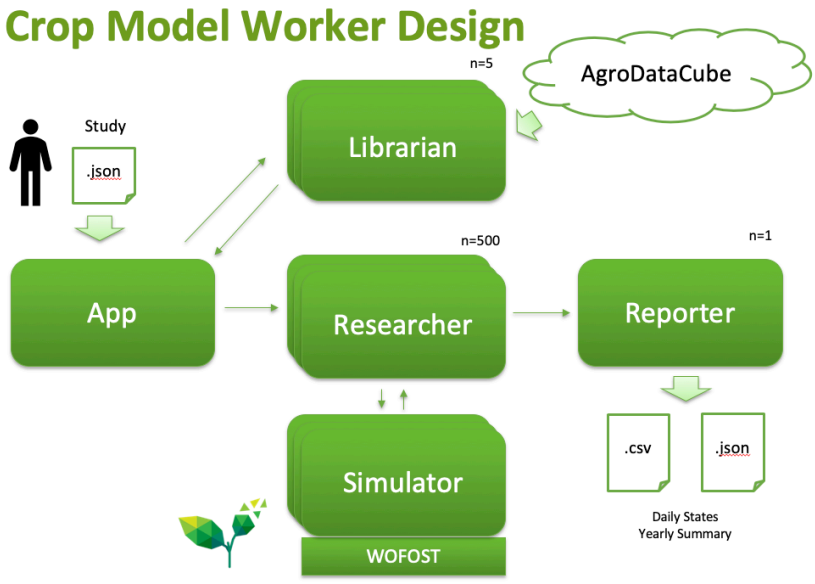

Fig. 2. Actors in the Worker algorithm. The user provides a description of a study in JSON format. The Librarians (n=5) smartly retrieves all required data from AgroDataCube and other data sources. Multiple Researchers (n=500) run WOFOST crop simulations in parallel via the Simulator. A single Reporter (n=1) collects and summarizes all results.

Retrieving data from the AgroDataCube is a clear bottleneck that technically should be solved. For now, data retrieved is cached and reused when possible, so after warm-up the algorithm optimizes for running similar crop simulations (e.g. in the same region with little variation in weather and soils). The output of the algorithm is either a CSV file with daily states of all calculated variables in the simulation, in case a single simulation is run, or a JSON file with a summary of the main variables for each simulation run. This keeps the total output manageable and should the detailed data for a specific simulation be needed it can easily be produced.

The 'scheduler' algorithm (**Fig. 3**) retrieves all IDs of crop fields from the AgroDataCube that match criteria specified by the user of the system, i.e. crop to process, year to simulate, spatial region the crop fields should be in. And some more technical parameters as well such as the preferred batch size, the maximum number of batches to process, and a timeout value for the total processing. These might be hidden after the system has been tuned and runs in a more operational state. Given the list of IDs the algorithm creates the batches, and for each one sends a WPS request to DataMiner to run the crop simulations algorithm. The scheduler then tracks the status of all algorithm executions in progress. When they all have finished all produced output is analyzed by the algorithm and an overall summary is created and made available in HTML format, giving the research a quick overview of the state of all crop simulations performed.

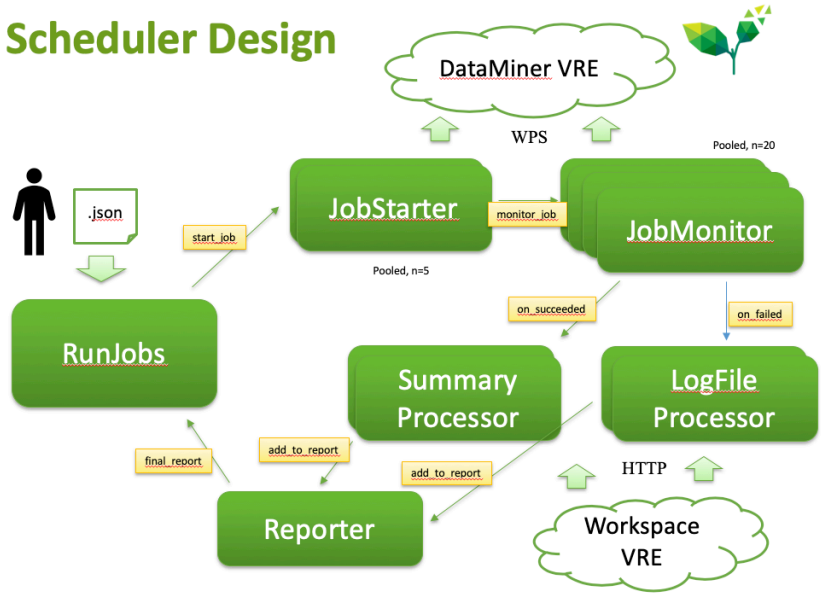

Fig. 3. Actors in the Scheduler algorithm. The user provides a description of a study in JSON format. A number of JobStarters send WPS requests to DataMiner to run the worker algorithm. Once started their execution progress is tracked by JobMonitors. Processing results (all the output files that DataMiner writes to the storage system) are then analyzed by LogFileProcessors and SummaryProcessors, that provide input to a Reporter that creates an overview.

Although improvements still need to be made and many optimizations are possible, it is clear that technically the solution works. Even in this stage and with a limited system of 6 fat nodes in the cluster, round-trip processing of crop simulations runs at about 50 simulations per second. Large workloads can easily be handled by (temporarily) adding more nodes to the cluster. And via WPS both types of algorithms can be called from other applications than the VRE. E.g. a custom web user interface, a mobile application, another WPS client such as QGIS (qgis.org), and so on. To showcase this within the AGINFRA PLUS project a dashboard visualization (see **Fig. 4**) is also being build, bringing together the input data from AgroDataCube and the output data from crop simulations.

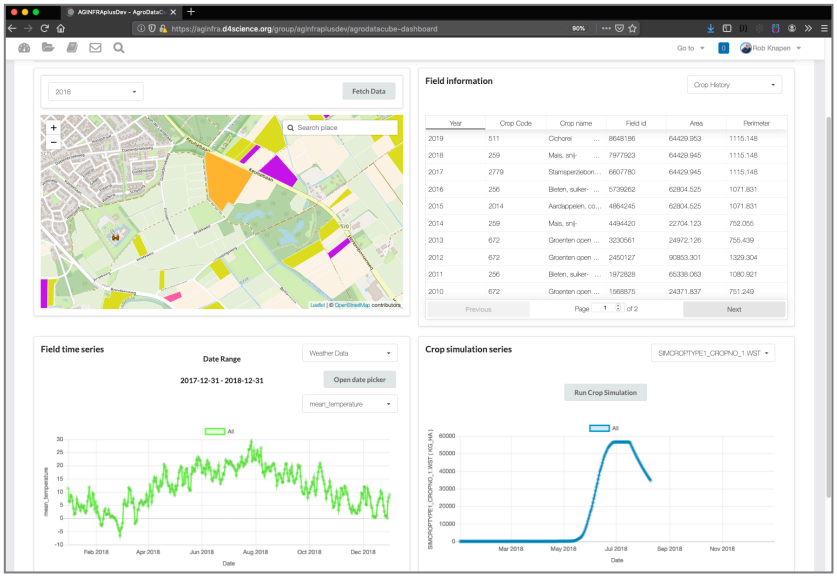

Fig. 4. The Crop Modelling Dashboard. It displays a map with crop fields (top left) from which a parcel can be selected. Data from AgroDataCube is then retrieved and field character-istics such as the crop, previous crops, soil types, etc. are displayed top ri right. In the bottom left weather and NDVI timeseries can be displayed, while the view on the bottom right side allows running a crop simulation for the selected field and visualization of the outputs, such as leaf area index, above ground biomass, soil moisture, harvest index, etc.

4 Conclusions

The described implementation allows running crop simulations in a standardized environment, with horizontal scalability. For larger workloads required computational time can be decreased by adding more computers to the cluster. Besides, the processes are

available via the standardized WPS interfaces and can thus be used in many applications. Or within VREs, tailored to specific user communities and usages. Further fine-tuning of the system is of course needed to turn the current proof-of-concept into an operational solution. With similar data as the contents of the AgroDataCube and proper calibration of the WOFOST-WISS model the same system can also be used for other regions.

Parts of the system will also be used in the Cybele EU H2020 project [4], which looks at the convergence of Cloud Computing services, such as D4Science, and High Performance Computing (HPC). This should make available even more compute power for running crop simulations, opening the door for doing even more advanced studies requiring ever increasing numbers of crop field level simulations to be run, e.g. yield forecasting at the field level based on detailed weather forecasts.

Naturally the real value in the end is to be able to better advice farmers and achieve a more efficient and future-proof agri-food system.

Acknowledgment

This work has received funding from the European Union's Horizon 2020 research and innovation programme under the AGINFRA PLUS project (grant agreement No 731001).

References

1. Akka framework homepage: https://akka.io, 30/09/2019.
2. Assante, M., Boizet, A., Candela, L., Castelli, D., Cirillo, R., Coro, G., Fernandez, E., Filter, M., Frosini, L., Kakaletris, G., Katsivellis, P., Knapen, R., Lokers, R., Mangiacrapa, F., Pagano, P., Panichi, G., Penev, L., Sinibaldi, F., Zervas, P. (2019) Realising a Science Gateway for the Agri-Food: The Aginfra Plus Experience. *International Workshop on Science Gateways.* **11**
3. Assante, M., Candela, L., Castelli, D., Cirillo, R., Coro, G., Frosini, L., Lelii, L., Mangiacrapa, F., Pagano, P., Panichi, G. (2019) Enacting Open Science By D4science. *Future Generation Computer Systems.* 10.1016/j.future.2019.05.063
4. CYBELE EU H2020 project homepage: https://www.cybele-project.eu, 30/09/2019.
5. D4science homepage: https://www.d4science.org, 30/09/2019.
6. De Wit, A., Boogaard, H., Fumagalli, D., Janssen, S., Knapen, R., Van Kraalingen, D., Supit, I., Van Der Wijngaart, R., Van Diepen, K. (2019) 25 Years of the Wofost Cropping Systems Model. *Agricultural Systems.* **168**, 154-167.
7. Dean, J., Ghemawat, S. (2008) Mapreduce: Simplified Data Processing on Large Clusters. *Communications of the ACM.* **51**, 107-113.
8. gCube Framework homepage: https://www.gcube-system.org, 30/09/2019.
9. AgroDataCube: A Big Open Data collection for Agri-Food Applications: **https://doi.org/10.18174/455759**, 30/09/2019.
10. Lokers, R., Knapen, R., Janssen, S., Van Randen, Y., Jansen, J. (2016) Analysis of Big Data Technologies for Use in Agro-Environmental Science. *Environmental modelling & software.* **84**, 494-504.
11. W3C Prov-O: The PROV Ontology specification: https://www.w3.org/TR/prov-o/, 30/09/2019.