Geophysical Research Abstracts
Vol. 20, EGU2018-10407-1, 2018
EGU General Assembly 2018

Flexible development and deployment of large scale data processing in MAGIC

Niels Drost (1), Maarten Plieger (2), and Wim Som de Cerff (2)

(1) Netherlands eScience Center, Amsterdam, Netherlands (n.drost@esciencecenter.nl), (2) Royal Netherlands Meteorological Institute, R&D Observations and Data technology Netherlands

The Metrics and Access to Global Indices for Climate Projections(MAGIC) project, part of the Copernicus Climate Change Service (C3S), is developing a system allowing users to visualise and analyse Petabytes of climate model data without having to download them to their own machine.

The MAGIC system consists of a backend service capable of running analysis on climate data, and a frontend to let users request analysis to be done. The compute backend uses ESMValTool, a system for running diagnostics on climate model data output, and exposes this as a standard WPS service. Diagnostics can be written in any programming language, with explicit support for Python, R, NLC, and Fortran, and out of the box support for a number of scientific libraries is available including s2dverification and Iris.

Scientists add diagnostics to ESMValTool, which in turn can be added to our infrastructure relatively easily, without too much manual effort for each diagnostic. The frontend builds on ADAGUC services, giving functionality such as user management, visualisation of results through WMS and OpenDAP, download of results, etc.

By deploying the backend close to the data, we greatly improve the performance. In this session we will show how the combination of containers, software packaging, and modular infrastructure, allows us to deploy the infrastructure needed easily on any number of resources, including servers and clouds. Together with partners within C3S we will run the final backend near one or more ESGF nodes, for maximum performance.

Our system design, with standard interfaces, also allows us to change parts of the system relatively easily. This in turn allows us to iteratively improve on our system with increasingly stable releases on average twice a year. In this way we can take lessons learned from our system into account while the project is still running, without re-doing the entire system from scratch. It also allows us for re-use of software, as any component we build can also be used in a different context more easily.