PROJECTS BASED ON THE WEB PROCESSING SERVICE FRAMEWORK BIRDHOUSE

Carsten Ehbrecht[a], Tom Landry[b], Nils Hempelmann[c]*, David Huard[d], Stephan Kindermann[a]

[a] German Climate Computing Center (DKRZ), Germany
[b] Computer Research Institute of Montreal (CRIM), Canada
[c] Deutsche Gesellschaft für internationale Zusammenarbeit (GIZ), Germany
[d] Ouranos, Canada

Commission IV, WG IV/4

KEY WORDS: Web Processing Service (WPS), Birdhouse, Climate Model Data, Earth Observation Data, Climate Change Impact, Extreme Weather Event Assessment, EU Copernicus Climate Change service (C3S), Research Platforms, Climate Services, PAVICS, Testbed, Big Data, Artificial Intelligence, PyWPS, Open Geospatial Consortium (OGC), Sustainable Development Goals (SDG)

ABSTRACT:

Birdhouse is a collaborative project open for the community to participate. It is a software framework containing a collection of Web Processing Services (WPS). The deployed algorithms are focusing on Earth Systems and environmental data processing with the philosophy of streamlining the software development and deployment. By supporting climate, earth observation and biodiversity data and processes, Birdhouse can be used in a wide array of Earth sciences projects and workflows. The core benefit of this project is to allow the seamless use of climate services developed by a diverse network of national meteorological offices, regional climate service providers, academics, not-for-profit research centers and private industry. As governments move toward open-data policies, there will be a need for analytical services that extract value out of the deluge of information. Using an interoperable software architecture, institutions can provide both data and services allowing users to process the data remotely from a laptop, instead of having to acquire and maintain large storage infrastructures.

1. INTRODUCTION

Birdhouse[1] provides a toolbox to setup a Web Processing Service (WPS) infrastructure for project specific use cases. It currently uses the Python implementation of the Web Processing Service OGC standard, PyWPS 4.0, but can support other WPS implementations as well. The original focus of Birdhouse was to support data processing as a service in the climate science community. But the technical concepts are not restricted to a specific community. Birdhouse components are organized into independent modules called ”birds”, often centered around specific themes or problematics. For instance, new birds were developed in projects focusing on extreme weather event assessments. To support the climate services, core GIS and geospatial operations such as subsetting, regridding, averaging or buffering are offered. Birdhouse reuses several FOSS4G components, including OpenClimateGIS, PyWPS, OWSLib, SNAP, GeoServer, GDAL, Shapely, ncWMS and THREDDS. This paper first describes Birdhouse main features, a brief history and major projects leveraging it. It finally provides an overview of future collaborative work in its institutional landscape.

1.1 General description

Birdhouse is designed as easy to install and individually combine the set of WPS tools as needed. It provides Buildout, Ansible and Docker for the deployment and uses Conda packages to maintain the software dependencies. Additional WPS clients, either command-line based or UI-based, allows for testing and demonstration. New templates are now provided to hatch a new WPS to support the deployment of PyWPS services for individual use cases. Birdhouse offers a security proxy for Web Processing Services (WPS) called Twitcher that provides access control using access tokens and X509 certificates.

The development of the framework is lead by the German Climate Computing Center (DKRZ) as a community driven software. DKRZ participates to the large European project Copernicus Climate Change Service (C3S) where Birdhouse is used as the backend for the climate projection data processing. Birdhouse is used by the Canadian institutes Ouranos and Centre de recherche informatique de Montréal (CRIM) to develop the Platform for the Analysis and Visualization of Climate Science (PAVICS). Birdhouse and its FOSS ecosystem is also advanced in current OGC Testbed for sponsors worldwide.

1.2 History

Birdhouse was initially designed to process climate model outputs in form of netCDF files and stored in the distributed climate data archive of the Earth System Grid Federation (ESGF). The Birdhouse framework takes its roots in the Collaborative Climate Community (C3-INAD) project (Kindermann et al., 2012) aiming to develop and deploy grid computing solutions in participating German data centers to provide services for the climate and climate impact scientific community. Its development was further advanced by inclusion of technical data quality checks (QC) of netCDF data (see section 1.3). QC processes were developed to check the climate model output files before archiving them, thus ensuring their correctness and fitness. Other QC processes targeting CF and ACDD file standards are also supported by the means of the Compliance Checker from Integrated Ocean Observing System (IOOS). Birdhouse is also moving to *harnessing WPS for sustainable development*, starting to cover recommendations and demands of international juridical Conventions of the

*Corresponding author – info@nilshempelmann.de
[1]Birdhouse documentation http://bird-house.github.io/

United Nations (UN). It is in line with the Paris Agreement and provides more and more processes to support the realization of the sustainable development goals (SDG).

1.3 Thematics being covered

Out of the history of Birdhouse the deployed algorithms are to provide services for the climate model and climate impact community. The core functions are focusing on Climate change assessment and climate impact studies (Hempelmann et al., 2018). Climate impact in particular is supported by the Atmospheric flow Analogues for Climate Change (A2C2) project [2] which aims to study rare and extreme weather events. While there are processes supporting the scientists and stakeholder in terms of climate change issues, the data-centers hosting the data archives are using Birdhouse modules for data management in terms of quality control (Jung et al., 2017). Beside the QCs on an technical aspect the quality of a climate model output can also be measured in terms of representing appropriate climatological phenomenons. For that, ESMValTool (Eyring et al., 2015) as a WPS service is going to provide processes for thematic climate model output quality check. For example the representation of El Niño in an output can be determined.

The extensive institutional landscape of Birdhouse and its FOSS ecosystem enables notable Artificial Intelligence (AI) possibilities. Collaborating institutions have already noted several Deep Learning projects on very big Earth Systems data (Kim, 2017, Dahmane et al., 2017). Notably, current work in PAVICS plans for Machine Learning applications for satellite imagery. The requirements for large scaled AI-enabled architectures heavily relies on next generation federated infrastructures and other geospatial Big Data projects (Percivall, 2017).

2. TECHNIQUES IMPLEMENTED

The Birdhouse software framework currently uses the PyWPS 4.0 Python implementation of the Open Geospatial Consortium (OGC) WPS standard. It is organized in a modular way (named after birds) and contains beside Python code some implementations of R and FORTRAN. Birdhouse depends on several geospatial libraries, such as OpenClimateGIS, GDAL and CDO for climate data.

The user friendly installation mechanism is realized by Conda packages, listed in environment files which enables an installation independently of the local server architecture. Since Birdhouse was originally designed to process climate model output there is a WPS module (malleefowl) to search and download data from the network of nodes of ESGF[3].

2.1 Deployment

The basic infrastructure for a WPS needs at least the following software packages: Nginx (Web-Server), Gunicorn (Python WSGI Application service), Supervisor (service monitoring) and PyWPS. The list of software dependencies may rise quickly for the processing codes served by the WPS (R, ESMValTool, OpenClimateGIS, GDAL, ...). Furthermore the complete installation should run on several Linux operating systems automatically. A default installation is done by a simple *make install* without reading lengthly documentation.

To achieve this goal we use the Conda package manager to handle the burden of dependency management on different Linux flavors. The installation of the WPS application is controlled by Buildout, a Python automation tool to assembly software. Buildout can be extended with *recipes*. Birdhouse provides recipes for the WPS application installation including templates for a customized configuration. Recently we have started to use Ansible for the deployment on the system level (deploy on remote hosts). A Docker image is automatically build for each WPS and available on Docker Cloud.

Birdhouse supports also to setup a new WPS (”bird”) from scratch by just answering a few question like the WPS service name. This is realized with a Python *cookiecutter* template. The template has the complete layout of a WPS and it can be customized by running the *cookiecutter*. Additionally, in OGC Testbed-13 and 14, PyWPS is being enhanced to run Application Packages. These applications aim to follow best practices from registries, catalogs, deployment and execution.

2.2 OWS Security Proxy Twitcher

OGC Services like WMS, WFS and WPS can be secured by using HTTPS but the standard does not define ways to control access to the service or parts of service offerings. In case of Web Processing Services (WPS) we are confronted with the need to restrict access of a process execution to users with the appropriate permissions. Twitcher provides access control for WPS services and can be extended for other OGC services as well.

Twitcher is a security proxy for WPS. The WPS itself does not need to be changed. Twitcher is a middleware component between the WPS client and the WPS server. All requests from the client are controlled by Twitcher. In case of a WPS service the *GetCapabilities* and *DescribeProcess* operations are not restricted. The *Execute* operation is blocked by Twitcher. The *Execute* operation can only be used if the WPS client provides a valid credential like an access token (via HTTP header) or an X509 certificate.

OGC is currently working on a new Web Service Security Standard. The concepts of this new standard are in line with the architecture of Twitcher. Twitcher will be adapted to this standard when it evolves. For instance, PAVICS introduced a new AuthZ module called Magpie. The module implements the Spring Security model and offers administrative tools to manage OWS permissions. Magpie is tightly integrated and complementary to Twitcher and has been described in OGC Testbed-13 EOC Cloud ER. Further work is conducted on federated security issues in the Earth Observation and Clouds thread of OGC Testbed-14.

2.3 Workflows

Workflow engines can be used to chain processes of Web Processing Services. Birdhouse provides some simple workflows using the Python *Dispel4Py* library (Filguiera et al., 2014). One workflow is to select and download climate data from federated ESGF data nodes. Depending on the data selection constraints (search facets) the data will be fetched from ESGF nodes and provided to a user specified WPS process, for example a Climate and Forecast (CF) quality check tool. This workflow is provided as WPS process itself and can be called by a WPS client directly without learning a new API.

The workflows are available in the *malleefowl* WPS provided by Birdhouse. In PAVICS, the workflow module was advanced to

[2] A2C2 Project: https://a2c2.lsce.ipsl.fr/
[3] Earth System Grid Federation https://esgf.llnl.gov/

include distributed execution and enhanced scripting capabilities. Exploration and investigation was conducted on workflow solutions in distributed computing initiatives in the ESGF ecosystem (Maxwell and Duffy, 2016, Fiore et al., 2013).

2.4 WPS Clients

Users can use WPS services directly using HTTP requests with *curl* or more comfortable using the *OWSLib* Python library. Birdhouse provides in addition a command-line client (Birdy) and web-UI (Phoenix). The main intention of these generic WPS clients is to use an arbitrary WPS server right away without remembering the WPS protocol details. These clients can be used in the development phase of a WPS as well as for demonstrations. Usually, new projects want to have their own specific web portal, like the PAVICS portal, which is customized to the project users and needs.

2.5 Earth Observation (EO) services

Recently, the Sentinel Application Platform (SNAP) provided by the European Space Agency (ESA) has been integrated into Birdhouse via its Python API called snappy as well as by means of a standardized Docker application packaging. SNAP (Zuhlke et al., 2015) allows for the rapid creation of new processes and execution graphs relying on satellite-based EO data. To further facilitate EO data access, the scihub.copernicus was used in a WPS.

In OGC Testbed-13, SNAP was integrated into an Application Package. The application was used as a general EO workflow/graph runner. It was requested through a WPS 1.0 server (PyWPS 4.0) in an hybrid cloud PubSub model. Other FOSS4G was used, including GeoServer (WMS,WCS), GDAL, Radarsat-2 Toolbox (RSTB), QGIS and OWSLib. Recommendations were provided in OGC TB-13 Cloud ER in regards to WPS 2.0 compatibility.

3. PROJECTS BASED ON BIRDHOUSE

Birdhouse is a follow up from the C3-GRID INAD project. Based on the ”lessons learned” Birdhouse has been built from the start to scale. Additional WPS services are easy to add and new servers are easy to deploy on various hardware architectures. Services can be modified or combined with others according to users needs, and independent climate service providers can interconnect their servers to benefit from the services and infrastructures of the community. The activity is already a small ecosystem of digital services implemented around the world. The current and upcoming projects are developing new features.

3.1 CP4CDS - Climate Projections for Climate Data Store

Copernicus is the European Union’s earth observation program. It provides data from multiple sources: earth observation, satellites and in-situ sensors. It focuses on several thematic areas like land, marine, atmosphere and climate change. The services and data are freely available to everyone. The intended users are policy-makers and public authorities.

The Copernicus Climate Change Service (C3S) offers a Service Portal that gives access to information for monitoring and predicting climate change to support adaptation and mitigation. The C3S Service Portal is hosted at ECMWF, UK.

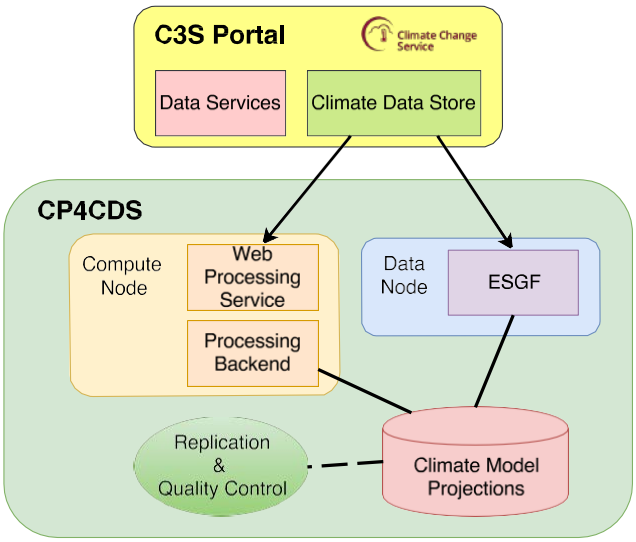

Figure 1. CP4CDS Architecture Overview

The Climate Data Store (CDS) contains the geophysical information needed to analyze the climate change indicators in a consistent and harmonized way. The store will provide both distributed data resources and computing facilities to the Copernicus Climate Change Service (C3S).

Birdhouse is used in the Copernicus sub-project CP4CDS[4], Climate Projections for the Climate Data Store. CP4CDS provides the required data and services for global climate projections (CMIP5) to the Climate Data Store (CDS). The data nodes consist of the Earth System Grid Federation (ESGF) software stack (THREDDS data catalogs, Solr search index) to provide climate data access. The compute nodes provide compute facilities using the Web Processing Service standard interface. CP4CDS is using Birdhouse components for automatic deployment of WPS with direct access to a climate data pool.

The CP4CDS nodes are a geographically distributed and highly available set of data and compute services. They are load-balanced across three sites, the leading European climate compute centers: CEDA, IPSL and DKRZ. To load-balance the compute requests at each site the PyWPS scheduler extension (Slurm, Grid-Engine) is used to delegate job execution to the cluster nodes available at the compute-centers.

Birdhouse provides the Software Deployment and Dependency Solution (SDDS) to manage and deploy processing codes into the distributed CP4CDS compute nodes. It uses Conda and Ansible to manage software dependencies and deployment of a WPS. Birdhouse provides in addition a template (using Python cookiecutter) to setup a new WPS for processing tools working on the climate projection data pool. The climate data processing is based on the Earth System Evaluation Tool (ESMValTool) by German Aerospace Center (DLR). The processes consist of Python and R routines to calculate metrics and multi-model products. The CP4CDS infrastructure is prepared to serve additional climate data (CORDEX, CMIP6, ...) and the corresponding processing modules.

3.2 PAVICS - a new CANARIE Research Software

CANARIE, Canada’s Advanced Research and Innovation Network, operates the national backbone network of Canada’s Na-

[4]CP4CDS Poster at ESGF F2F https://bit.ly/2vXE95v

tional Research and Education Network (NREN). It supports the development of research software tools[5] and infrastructure for research data management. CANARIE funded the Platform for the Analysis and Visualization of Climate Science (PAVICS) developed conjointly in Canada by Ouranos and CRIM. PAVICS (St-Denis et al., 2017) is composed of a set of WPS and user-facing tools meant to speed up the analysis of climate data and the creation of climate scenarios for impact studies. The platform and its key components are monitored and tracked in the CANARIE Software Registry.

PAVICS[6] makes extensive use of several core birds of the framework. It advances Twitcher proxy and load balancing capabilities. It introduces authorization mechanisms in a new component called Magpie. PAVICS allows viewing and selection of features through WFS as inputs to WPS. PAVICS also introduces distributed workflow execution, based on Dispel4Py, in the Malleefowl bird.

Figure 2. The PAVICS Research Platform

Most recent work in PAVICS includes packaging and tooling of advanced hydroclimatology services. New features will also include processing of very high resolution imagery and synthetic aperture radar (SAR) imagery, including RADARSAT-2 and Sentinel missions. Additional GIS capabilities in the user interface will allow annotation of features and improved imagery support. Deep Learning and Natural Language Processing (NLP) models, tools, services and best practices will be progressively published and promoted in the Birdhouse and ESGF ecosystems. Future releases PAVICS components will seek interoperability with C3S Portal and CP4CDS.

3.3 OGC Testbeds - Earth Observation and Clouds

CRIM was in charge of delivering a system implementation and contributing to an engineering report in OGCs Testbed-13[7] Earth Observation Clouds (TB-13 EOC) thread. The deliverable aimed to advance cloud API interoperability and application portability as key elements in hybrid cloud computing research. The novel PyWPS job scheduler extension to High Performance Computing (HPC) component, developed by DKRZ, was reused. An hybrid cloud scheduler for PyWPS based on Celery and RabbitMQ was developed by CRIM. Recommendations on the use of OWS Context standards for application packaging, as well as potential WPS 2.0 support, were provided to OGC in (Chen et al., 2018). Work is currently conducted by CRIM in the EOC thread of OGC Testbed-14, where key findings of TB-13 are further advanced into ESA Thematic Exploitation Platform (TEP) architecture[8]. Whenever possible, source code will be released under FOSS licenses compatible with PAVICS and Birdhouse.

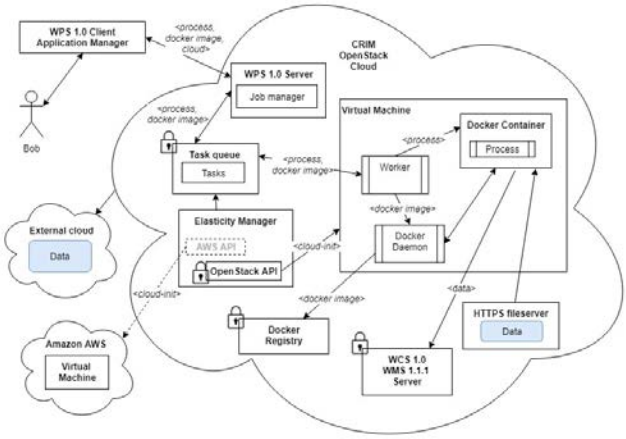

Figure 3. Cloud architecture contributed by CRIM in OGC Testbed-13 Cloud ER

4. INSTITUTIONAL LANDSCAPE

The mentioned projects are already showing the heterogeneity of institutional landscape. So far the Birdhouse software framework brought together climate service centers, national laboratories and research centers in E.U. (DKRZ, LSCE, IPSL, CEDA, ECMWF, BADC, ESA), Canada (CRIM, Ouranos, CCSC, PCIC, CGDI, NRCan) and in the U.S. (NOAA, NASA, LLNL). Among these stakeholders is shared a massive amount of FOSS4G.

The project will be part of the backend for the European C3S project and is currently entering the Copernicus consortium. Birdhouse will also be a key component of similar yet smaller scale deployments on Canada's cyberinfrastructure. In parallel, the project was applied in OGC testbeds for the benefit of ESA, NRCan and other international sponsors. OGC members can participate or observe in OGC Innovation Program (IP) initiatives, advancing the maturity of their FOSS components. Currently, CRIM, Ouranos, DKRZ and several national laboratories and agencies in ESGF are either participants or sponsors at OGC, including ESA that very recently joined the rank of strategic members. The project has therefore excellent chances to be used in scientific and technological transfer between Canada, Germany and other industrialized or developing countries.

The interoperability of the services offers a great solution to less and least developed countries with low digital bandwidth and limited high performance computing infrastructures. More importantly, in the spirit of the United Nations (U.N.) General Assembly Agenda 2030[9] for Sustainable Development "*leave no one behind*", these countries can leverage existing infrastructures elsewhere to offer their citizens custom-designed climate services and practical tools to close the digital gap between countries. These benefits are driving discussions about its use at the German

[5] CANARIE Research Software Program https://www.canarie.ca/software/platforms/

[6] PAVICS official documentation https://ouranosinc.github.io/pavics-sdi/

[7] OGC Testbed 13 publications http://www.opengeospatial.org/pub/Testbed13/index.html

[8] Exploitation Platforms Open Architecture http://bit.ly/2m1Gnsm

[9] U.N. 2030 Agenda for Sustainable Development http://bit.ly/1Y3D3sN

Cooperation (GIZ) development agency to provide front-ends for users in less and least developed countries. In Canada, this digital divide is observed in remote and often poorly deserved regions, including indigenous and native communities. An important number of actions are undertaken in Smart Cities[10] initiatives, involving all levels of government; national, regional (provinces, *comtés*) and municipal (cities, towns, *arrondissements*).

WPS is enabling big data processing launched by users located in low bandwith internet regions and not requiring high-tech computer power. For that reason, less and least developed countries are able to perform the same analytics like their colleagues in developed countries. ESA are directly supporting the Development Assistance Committee (DAC) with portal functionality development. Joint workshops are taking place to exchange experience of ESA and development cooperation agencies like GIZ. On the other hand, ESA is supporting the OGC testbeds to get the appropriate portal functionality realized. Several R&D programs managed by space agencies (DLR, CSA, NASA, ESA) enable creation of next generation advanced data products and applications. Testbeds are realized with contribution of the Birdhouse consortium (see section 3.3), while notable research outputs are produced by the researchers it aims to serve.

5. CONCLUSION

In this paper we outlined the technical concepts of Birdhouse and its usage in climate science infrastructure projects like Copernicus CP4CDS (E.U.) and PAVICS (Canada). The intention of Birdhouse is to provide a good starting point for projects to setup and use Web Processing Services for their own use-cases. The *real* work still needs to be done in the projects themselves, for instance implementing the algorithms and developing web-based user interfaces. Birdhouse is *not* yet another processing framework. Rather it combines existing frameworks and tools to provide an adaptable solution. In some cases, like for the OWS security proxy, we provide our own implementation, like Twitcher, when there is no common tool yet available. In general, we actively support the development of already existing tools like OWSLib and PyWPS.

Birdhouse focuses on OGC standards and service interfaces, mainly the Web Processing Service standard. Web Processing Services can support the sharing of resources on climate change information. Birdhouse components can be combined, replaced or one may just use the existing projects as inspiration. Some of the key components of Birdhouse are used in standardization, interoperability and best practices efforts at OGC. Throughout the birdhouse development, we realized that the task is not only to combine the technical components but also to bring together people of different backgrounds and parts of our world. Only by joining forces and efforts, *ensemble*, can we succeed to make the world a bit better. Birdhouse and its siblings projects are publicly available and open for contribution.

ACKNOWLEDGEMENTS

The European authors acknowledge that Birdhouse development was funded by European Union (Copernicus) and the German Helmholtz Association (LSDMA). The Canadian authors acknowledge that part of this work was funded by the Ministère de l'Économie, de la Science et de l'Innovation (MESI) of Québec, by CANARIE and by OGC.

[10] Infrastructure Canada - Smart Cities Challenge http://www.infrastructure.gc.ca/cities-villes/index-eng.html

REFERENCES

Chen, C., Landry, T., Sun, Z. and Zhang, C., 2018. OGC Testbed-13: Cloud ER. http://docs.opengeospatial.org/per/17-035.html. OGC 17-035.

Dahmane, M., Foucher, S., Beaulieu, M., Bouroubi, Y. and Benoit, M., 2017. *The Potential of Deep Features for Small Object Class Identification in Very High Resolution Remote Sensing Imagery*. Springer International Publishing, Cham, pp. 569–577.

Eyring, V., Righi, M., Lauer, A., Evaldsson, M., Wenzel, S., Jones, C., Anav, A., Andrews, O., Cionni, I., Davin, E. L. et al., 2015. ESMValTool (v1.0) - a community diagnostic and performance metrics tool for routine evaluation of earth system models in cmip. *Geophysical Model Development Discussions* 8(9), pp. 7451–7661.

Filguiera, R., Klampanos, I., Krause, A., David, M., Moreno, A. and Atkinson, M., 2014. Dispel4Py: A Python Framework for Data-Intensive Scientific Computing. In: *2014 International Workshop on Data Intensive Scalable Computing Systems*, pp. 9–16.

Fiore, S., D'Anca, A., Palazzo, C., Foster, I., Williams, D. and Aloisio, G., 2013. Ophidia: Toward Big Data Analytics for eScience. *Procedia Computer Science* 18(Supplement C), pp. 2376 – 2385. 2013 International Conference on Computational Science.

Hempelmann, N., Ehbrecht, C., Alvarez-Castro, C., Brockmann, P., Falk, W., Hoffmann, J., Kindermann, S., Koziol, B., Nangini, C., Radanovics, S., Vautard, R. and Yiou, P., 2018. Web Processing Service for climate impact and extreme weather event analyses. Flyingpigeon (version 1.0). *Computers & Geosciences* 110(Supplement C), pp. 65 – 72.

Jung, C., Meyer, J. and Streit, A. (eds), 2017. *Helmholtz Portfolio Theme Large-Scale Data Management and Analysis (LSDMA)*. KIT Scientific Publishing, Karlsruhe. 46.12.02; LK 01.

Kim, S., 2017. Deep learning application for community machine learning. https://esgf.llnl.gov/media/2017-F2F/Day3/Deep-Learning-Kim.pdf.

Kindermann, S., Schintke, F. and Fritzsch, B., 2012. A collaborative data management infrastructure for climate data analysis. *Geophysical Research Abstracts* 14(EGU201), pp. 10569.

Maxwell, T. and Duffy, D., 2016. The Climate Data Analytic Services (CDAS) Framework. *AGU Fall Meeting Abstracts*.

Percivall, G., 2017. Big Geospatial Data - an OGC White Paper. http://docs.opengeospatial.org/wp/16-131r2/16-131r2.html. document 16-131r2.

St-Denis, B. G., Landry, T., Huard, D., Byrns, D., Chaumont, D. and Foucher, S., 2017. PAVICS: A Platform for the Analysis and Visualization of Climate Science - adopting a workflow-based analysis method for dealing with a multitude of climate data sources. In: *American Geophysical Union Fall Meeting*, New Orleans, Louisiana.

Zuhlke, M., Fomferra, N., Brockmann, C., Peters, M., Veci, L., Malik, J. and Regner, P., 2015. SNAP (Sentinel Application Platform) and the ESA Sentinel 3 Toolbox. In: *Sentinel-3 for Science Workshop*, Vol. 734, p. 21.

Revised April 30th 2018