DASF: Let's make scientific software available on the web

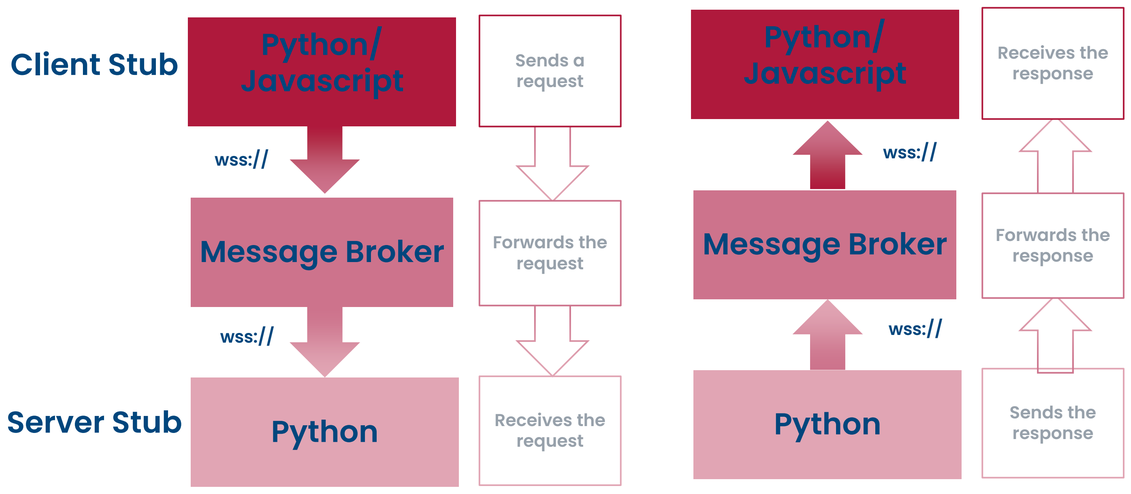
Philipp S. Sommer, Linda Baldewein
Helmholtz-Zentrum Hereon, Helmholtz Coastal Data Center, Kontakt: philipp.sommer@hereon.de

The Data Analytics Software Framework

Summary
The success of scientific projects increasingly depends on using data analysis tools and data in distributed IT infrastructures. Scientists need to use appropriate data analysis tools and data, extract patterns from data using appropriate computational resources, and interpret the extracted patterns. Data analysis tools and data reside on different machines because the volume of the data often demands specific resources for their storage and processing, and data analysis tools usually require specific computational resources and run-time environments. The data analytics software framework *DASF*, which we develop in Digital Earth and DataHub, provides a framework for scientists to conduct data analysis in distributed environments.

Statement of need
The data analytics software framework *DASF* supports scientists to conduct data analysis in distributed IT infrastructures by sharing data analysis tools and data. For this purpose, DASF defines a remote procedure call (RPC) messaging protocol that uses a central message broker instance. Scientists can augment their tools and data with this protocol to share them with others. *DASF* supports many programming languages and platforms since the implementation of the protocol uses secured Websockets (WSS). It provides two ready-to-use language bindings for the messaging protocol, one for Python and one for the Typescript programming language. In order to share a python method or class, users add an annotation in front of it. In addition, users need to specify the connection parameters of the message broker. The central message broker approach allows the method and the client calling the method to actively establish a connection, which enables using methods deployed behind firewalls.

Remote Procedure Call



Automated API generation

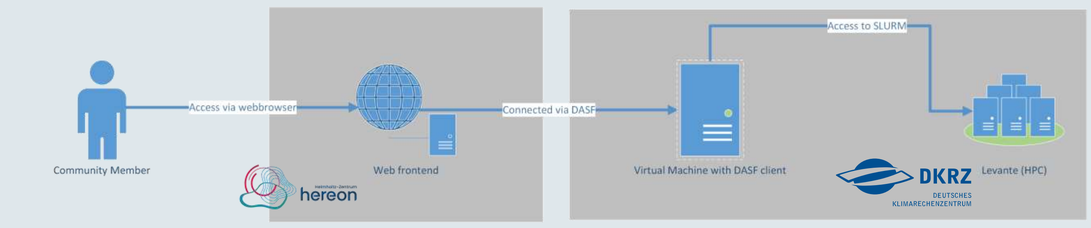
```python
from demessaging import main

    You, seconds ago • Uncommitted changes
def compute_sum(da: DataArray) -> DataArray:
    """Compute the sum over a data array.

    Parameters
    ----------
    da : DataArray
        The input data array

    Returns
    -------
    DataArray
        The sum of the data array
    """
    return da.sum()


if __name__ == "__main__":
    main(topic="hello_world")
```

Backend module: Example for a backend module (server stub)

```python
def compute_sum(
    da: demessaging.types.xarray.DataArray,
) -> demessaging.types.xarray.DataArray:
    """
    Compute the sum over a data array.

    Parameters
    ----------
    da : DataArray
        The input data array

    Returns
    -------
    DataArray
        The sum of the data array
    """
    request = {
        "member": {
            "func_name": "compute_sum",
            "da": da,
        }
    }

    model = BackendModule.parse_obj(request)
    model.compute()

    return model.member.func_returns  # type: ignore
```

Client module: Automatically generated python module (client stub) that connects to the message broker to compute the results on the remote server

Showcase 1: WebPEP



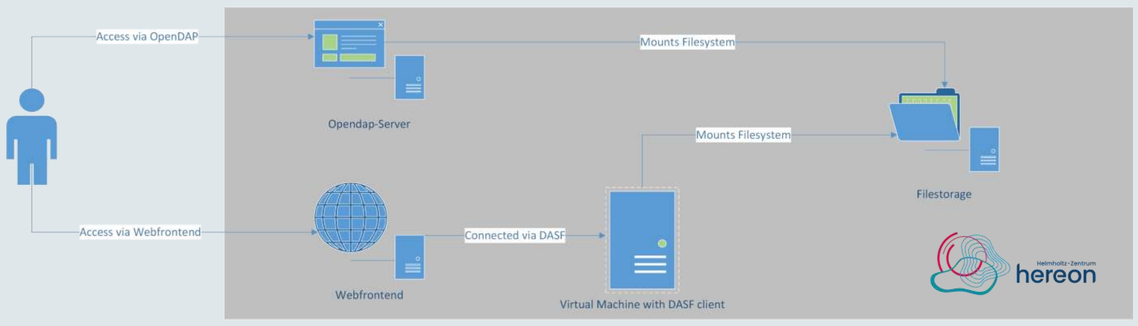
Generation of Input Data for COSMO-CLM

Scope of DASF: web interface to a scientific software that generates external parameters for a regional climate model.

Web frontend	Django-based Community Portal with implemented Message Broker. Includes a permission system and makes the website available to community members.
Client machine	Virtual machine at the DKRZ running the DASF client that securely connects to the website via WSS. Has only access to the scheduler and specific folders on Levante.
Levante	Supercomputer at the German Climate Computing Center (DKRZ) where the scientific software to generate the data is installed.

https://hcdc.hereon.de/clm-community

Showcase 2: Riverplume Workflow



Processing data for a static webfrontend

Scope of DASF: The Riverplume Workflow has a static frontend that needs to process data that is additionally available via OpenDAP. Using DASF makes the file access faster than OpenDAP alone.

Web frontend	Static webserver (here Django-based for additional permissions) with Message Broker.
DASF Client machine	Virtual machine at Hereon with direct access to files that are also available via OpenDAP. Connects to the static webfrontend via DASF and is used for interactive exploration of the data.
OpenDAP Server	Webserver to access the raw data using a standard protocoll.
Filestorage	NFS Storage with the raw data for the workflow that is mounted on the OpenDAP server and the DASF client machine.

https://digitalearth-hgf.de

Funding
This framework has been developed in a joint effort of DataHub and Digitial Earth initiatives within the Research Centers of the Helmholtz Association of German Research Centres, HGF.

References
- Eggert et al., (2022). DASF: A data analytics software framework for distributed environments. Journal of Open Source Software, 7(78), 4052, https://doi.org/10.21105/joss.04052
- Rabe, Daniela; Eggert, Daniel; Wichert, Viktoria; Abraham, Nicola (2022): The River Plume Workflow of the Flood Event Explorer: Detection and impact assessment of a river plume. GFZ Data Services. https://doi.org/10.5880/GFZ.1.4.2022.006

DASF Paper

Helmholtz-Zentrum **Hereon** • Max-Planck-Straße 1 • 21502 Geesthacht I Germany • T +49 4152 87-0 • contact@hereon.de • www.hereon.de
Contact: Philipp S. Sommer • T +49 4152 87-2126 • philipp.sommer@hereon.de