Data Analysis in Bridge of Data

Wojciech Artichowicz[1*], Krzysztof Drypczewski[2], Jerzy Proficz[2]

[1] Faculty of Civil and Environmental Engineering, Gdańsk University of Technology (11/12 Gabriela Narutowicza Street, Gdańsk, Poland)
[2] Center of Informatics Tri-Citry Academic Supercomputer and network (CI TASK), Gdańsk University of Technology (11/12 Gabriela Narutowicza Street, Gdańsk, Poland)

* Correspondence author: wojciech.artichowicz@pg.edu.pl; ORCID: 0000-0002-7582-1052

Abstract

The chapter presents the data analysis aspects of the Bridge of Data project. The software framework used, Jupyter, and its configuration are presented. The solution's architecture, including the TRYTON supercomputer as the underlying infrastructure, is described. The use case templates provided by the Stat-reducer application are presented, including data analysis related to spatial points' cloud-, audio- and wind-related research.

Keywords: Jupyter, Kubernetes, Bridge of Data, Big Data, data analysis, interactive computing, Stat-reducer

https://doi.org/10.34808/x55q-sz53_dyr_roz2

Introduction

One of the main goals of the Bridge of Data project is to provide a repository filled with research data complying with an Open Access policy. The project is being implemented by three Gdańsk universities: Gdańsk University of Technology (leader), University of Gdańsk and Medical University of Gdańsk. These organisations perform a wide range of research and are involved in data gathering and dissemination.

The repository itself is being developed by two teams from the Gdańsk University of Technology: Centre of IT Services (CUI) and Centre of Informatics Tricity Academic Supercomputer & Network (CI TASK). The former focuses on the front-end components, such as the user interface and input data validation, and the latter focuses on the backend, including data storage and analysis.

CI TASK provides network and computes services for Pomeranian researchers, including all of the public universities in the region. These services are based on advanced ICT (Information and Communication Technology) infrastructure with the TRYTON

supercomputer (Krawczyk, Nykiel and Proficz, 2015). This supercomputer is based on the cluster architecture. It consists of about 1600 compute nodes, each containing two processors (Intel Xeon Processor E5 v3, 2.3GHz, Haswell), with 12~physical cores (24 logical, based on Hyperthreading technology) and 128 GB RAM. Its total compute power is 1.48 PFLOPS, and its total weight is over 20 metric tons – 40 racks.

The main contribution of the chapter is an overall description of data analysis tools deployed within the Bridge of Data project. It presents the proposed services and typical examples of their usage, including a description of a spatial points' cloud-, audio- and wind-related research, which shows the extended capabilities of the proposed tools. The examples are prepared and documented to be easily reused and applied for other research problems.

Moreover, technical details of the proposed solution are described, especially the Jupyter platform dedicated for easy-to-use graphical user interface (GUI) based on the web technologies. This open-source solution supports several programming languages, including Python, Scala and R and provides an easy access to several important data analysis frameworks such as Apache Spark.

The data analysis within the scope of the Bridge of Data project is performed using TRYTON, within its separate (sub-)system managed by the OpenStack (Rosado and Bernardino, 2014) and Kubernetes (Bernstein, 2014) cloud platforms. Access to the computing power is provided by the Jupyter computing platform (Perkel, 2018), which provides a frontend console for researchers based on Python and other scripting languages. Support for Big Data processing, including MapReduce and beyond, is provided by the Apache Spark (Zaharia et al., 2016) analytics engine. The data used in this solution can be fetched from the Bridge of Knowledge repository or provided from external sources. Within the project's scope, Jupyter was enriched with template use cases – the Stat-reducer application, available for the users as open-source.

The following section presents the Jupyter configuration used to provide services supporting the data analysis, including its architecture and deployment on the CI TASK TRYTON supercomputer. The following section presents the Stat-reducer application, including the template use cases. The document is concluded with the final remarks and possible future directions of the works.

Services for data analysis

Jupyter is an open-source computing platform that enables cooperation with various technologies (libraries, programming languages, etc.). From the user's point of view, Jupyter provides a web application that is an engine for interactive task submission, data processing, and presentation of results. The most crucial element of the Jupyter Project is Jupyter Notebook, i.e., documents that define the tasks to be performed. A notebook can consist of many elements: hypertext, source code in multiple languages (Python, R, Scala, etc.), and presentation modules with associated data (tables, pictures, charts, etc.). Jupyter provides a rich programming interface that allows Notebooks to be easily expanded with additional functionalities (Ragan-Kelley et al., 2014), (Perkel, 2018).

The individual elements of a Notebook are commands which run in an interactive REPL (read-eval-print loop) console. Then, commands are passed to the JupyterKernel, which processes the user requests (executes code, communicates with a database, sends commands to external systems, etc.) and returns the response.

The JupyterLab project is responsible for the presentation of Notebooks to the user. It provides a web application that serves as an integrated development environment (IDE), equipped with a text editor, file browser, and the ability to control the flow of task execution (Kluyver et al., 2016).

JupyterHub is a Notebook server designed to make the Jupyter environment available to users over the Internet. It allows for building a fully integrated system: JupyterLab, access to storage space (Notebooks, data scripts, etc.), the configuration of individual kernels, authorisation and authentication system, etc. so that users do not have to carry out the local installation process themselves. Due to the ability to integrate JupyterHub with container technologies, it is possible to separate the environment of individual users completely and easily scale the system.

As part of the Bridge of Data project, a Jupyter-based computing environment was launched on the TRYTON supercomputer (Krawczyk, Nykiel and Proficz, 2015) located in CI TASK. Fig. 2.1 presents the architecture of the solution.

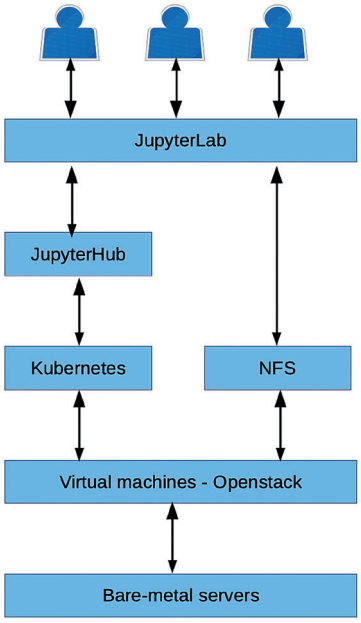

Fig. 2.1. The system architecture of the analysis system

The proposed system was launched in a private cloud managed by Openstack. In compliance with the DevOps methodology, the creation of virtual machines is fully automated with Ansible.

Containerisation is an operating system-level virtualisation technique that can isolate a user space within a single system kernel (kernel) (Boettiger, 2015). With container orchestration tools such as Docker, it is possible to deliver the application along with the entire environment (dependencies, libraries, configuration, etc.), as shown in Fig. 2.2. This approach increases application portability, enables more efficient use of resources (in comparison with virtual machines), and facilitates the use of continuous code integration and delivery (CI / CD) techniques.

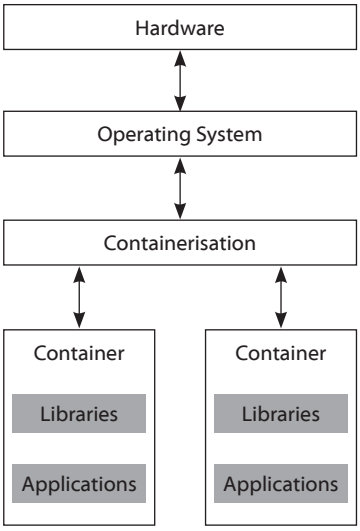

Fig. 2.2. Containerisation schema of the analysis system

Due to the high complexity of the system, its modules were separated using containers. Kubernetes (Bernstein, 2014) was selected as the container orchestration platform. It provides functionality to control the allocation of resources (per container/application/user), schedule the execution of tasks (allocation of pods to machines), adds support for network traffic between containers, application monitoring and automatic response to errors (Brewer, 2015). The computation run in Kubernetes is divided into 1) containers, 2) pods: groups of cooperating containers with a shared network, 3) services: groups of pods that create an application.

The following virtual machines were used for the Kubernetes deployment: 1) networking (i.e., ingress / egress controllers): 2 × 2 CPUs, 5 GB RAM, 20 GB SSD, 2) management: (Kubernetes services) 3 × 4 CPUs, 10 GB RAM, 40 GB SSD, 3) workers (services and pods runtime): 3 × 8 CPUs, 20 GB RAM, 80 GB SSD.

JupyterHub is deployed on the Kubernetes platform; the installation and configuration processes are performed with Ansible (automation) and Helm ("package manager" for Kubernetes). JupyterHub prepares a separate environment for each user based on a JupyterLab container. It is launched as a new pod available only for the current session

duration. To extend the functionality, a custom version of the JupyterLab installation has been prepared with the following characteristics:

JupyterKernels: Python, Scala, R, access to the standard Ubuntu Linux environment (bash, gcc, vi, etc.), possibility to install modules with pip and conda, up to 16 GB RAM per user/container, support for Apache Spark (Zaharia et al., 2016) analytics engine – deployed for Big Data processing, including map-reduce and other more complex programming models, preconfigured modules for data science and machine learning.

The data in each user's home directory is persistent, i.e. it is stored between logins. Access to the data is handled by an NFS server installed on a virtual machine, which has access to a 500 GB Ceph volume. Each user can use up to 10 GB of disk space.

User authorisation and authentication are handled by Keycloak, which provides a single sign-on (SSO) system for all CI TASK services. Thanks to the use of Keycloak, each JupyterHub instance can assign access and permissions to individual users and groups. Network traffic between users and the Notebook server is TLS encrypted and uses certificates obtained from Let's Encrypt.

The current version of the aforementioned software can be easily modified to include changes such as adding new modules and JupyterLab extensions, increasing the available computing resources (instance CPU, RAM and disk space), and scaling the Kubernetes cluster up or down.

Stat-reducer – use cases templates

The Stat-reducer functionality offered as part of the Bridge of Knowledge (MOST Wiedzy) platform is not a typical service delivered in the form of application functionality. The Stat-reducer service provides the Bridge of Knowledge platform users with a collection of Jupyter notebooks presenting complete examples of analyses of various types: from audio/video data, through environmental data, to the analysis of three-dimensional scans.

The essence of the functionality of Stat-reducer is the presentation of the capabilities of the Bridge of Knowledge platform in terms of processing and analysing data of various types:

loading and processing audio and video data, e.g. making periodograms and histograms of sound frequencies; point cloud analysis, e.g. convex hull determination and analysis of the principal components of the object; analysis and generalisation of tabular data, e.g., making histograms, determining the characteristic statistics of a dataset, etc.

These possibilities are presented in the form of Jupyter notebooks. Thanks to this, the user can easily use the code snippets from the notebooks. It can be said that Stat-reducer is documentation in the form of ready-to-use code. In the following subsections, some examples of Jupyter notebooks are provided as part of the Stat-reducer functionality.

Analysis of spatial data in the form of a point cloud

The data used in this example results from a laser scanning process of a fuel tank object. The coordinates of the points are saved in a text file in which the columns are separated by an empty character. The first step in the analysis is importing the data into memory. For this purpose, it is most convenient to use the Pandas module of the Python language, the basic object of which is a so-called data frame. This module offers the functionality of importing and exporting data from many different formats (SQL, CSV and text files, HDF5, JSON, etc.).

The second step of the analysis is the computation of the basic characteristics of the point cloud, such as the mean, standard deviation, minimum, maximum and quartiles. The easiest way to do the above calculations is to use the describe () method of the Pandas data frame.

The third step is to visualise the set and histograms of each x, y and z coordinates. For this purpose, the data frame method called hist() was used. The spatial plot of the point cloud presented in Fig. 2.3 was made with the Matplotlib library, whereas the coordinate histograms are depicted in Fig. 2.4.

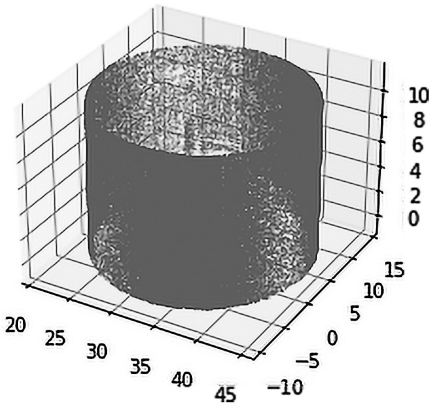

Fig. 2.3. Visualisation of the point cloud

Another issue raised in the analysis was the determination of the principal components of the set and their graphical presentation. To calculate the principal components, the data was centred, and then the PCA class of the sci-kit-learn library was used, which offers many basic machine learning algorithms. The result of the calculations is presented in Fig. 2.5.

The last element of the analysis was the creation of a convex hull of the set and then saving it in the STL format, which enables the file to be processed in software dedicated to preparing objects for 3D printing. The convex hull was determined using the ConvexHull function from the Scipy library (scipy.spatial). Convex hull export was performed using the NumPy-stl Python library. The analysis output is presented in Fig. 2.6, which shows the original set of points and its convex hull.

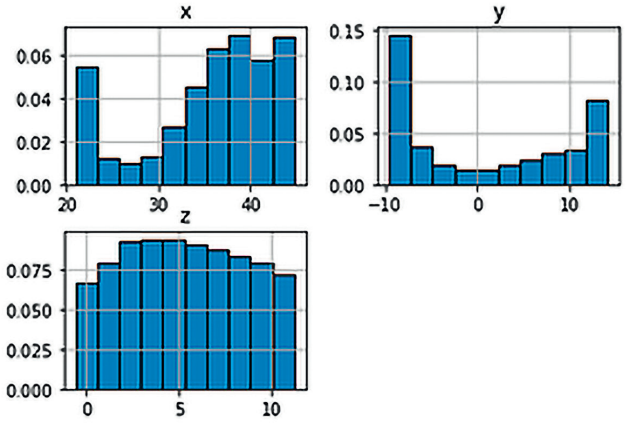

Fig. 2.4. Histograms of the point cloud coordinates

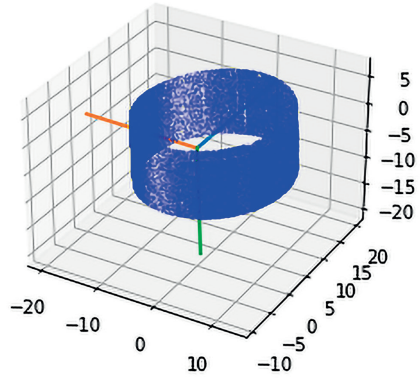

Fig. 2.5. Centred dataset (points) and its principal components (lines)

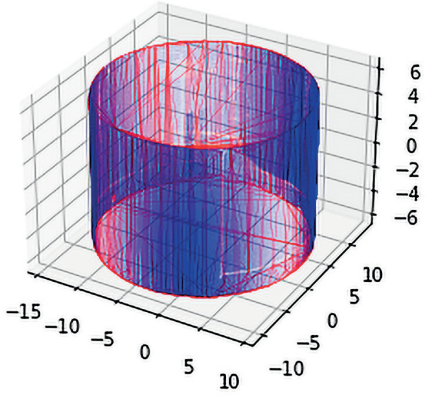

Fig. 2.6. The plot of the point cloud (blue) and its convex hull (red)

Audio analysis

The data used in this example consists of video files with sound, which contain recordings of bees flying into a hive. The focus of the analysis presented in this example is a visualisation of the audio present in the video files.

The first step in the analysis was to extract the sound from the video files and save them as .wav files. There are many possible approaches to this problem. However one of the simpler ones is to use the moviepy library, which offers the required functionality. Using the functionality of the os module of the standard Python library and the moviepy library, the sound was extracted from each of the video files and saved to disk as a WAV file. Then, the functionality of the Scipy library (scipy.io) was used to load the audio channels present in the files. Again, using the Scipy library (scipy.signal) and the Matplotlib library, a function visualising the sound in the waveform plots, periodograms and frequency histograms was developed. To make the exploratory analysis of many files convenient, an interactive widget in the form of a drop-down list was instantiated from the ipywidgets module. Each entry in the list refers to an audio file. The functionality was easily achieved by the interact function from the ipywidgets module, which was used as a decorator, to the visualising function. Finally, the created Jupyter Notebook document allows an audio file to be selected for which the abovementioned graphs are to be made. The effect of the abovementioned activities is the Jupyter Notebook document, including an interactive drop-down widget, which is presented in Fig. 2.7.

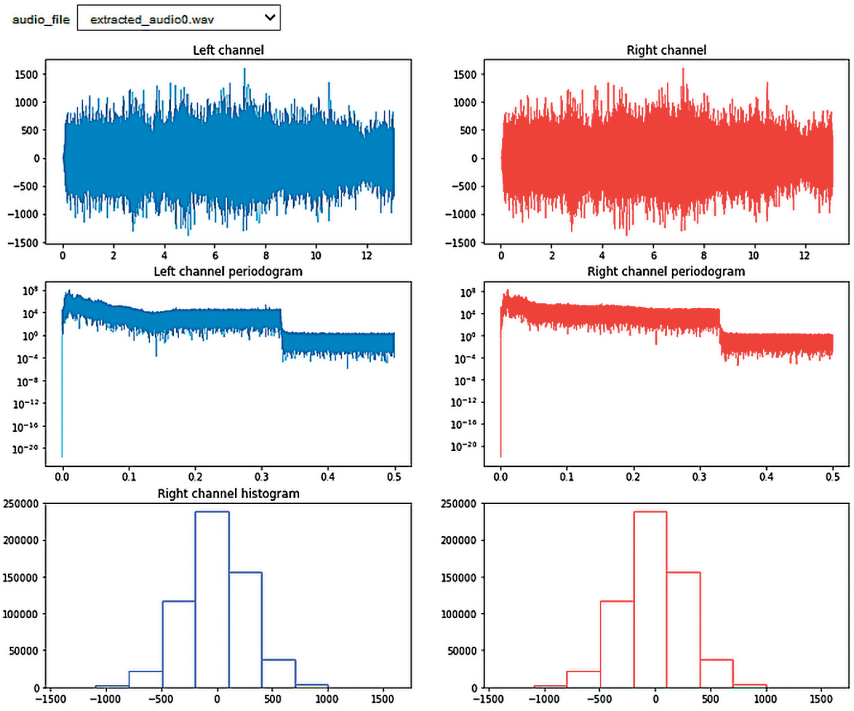

Fig. 2.7. Interactive Jupyter Notebook document for the graphical analysis of the audio files

Wind data analysis

The data used in this example relate to wind speed measurements at the Nowa Pasłęka location, made in 2008–2017 by the IMGW. The analysis presented here is a state-of-the-art analysis of the wind data.

Raw data containing information about the date and time of measurement, wind speed and wind direction is stored in an MS Excel© file. The most convenient way to handle the data is to read it into a Pandas data frame using the read_excel() function.

After reading the data into the data frame, it is necessary to check whether the data contains incorrect values. Such functionality is provided by the data frame object in the form of the isna() method, which identifies rows with incorrect values. Deleting rows containing incorrect data can be done using the Pandas data frame method called dropna().

After cleaning the data of incorrect values, a plot of the wind speed and direction is usually made. For this purpose, the Matplotlib library was used. The wind time-series is shown in Fig. 2.8.

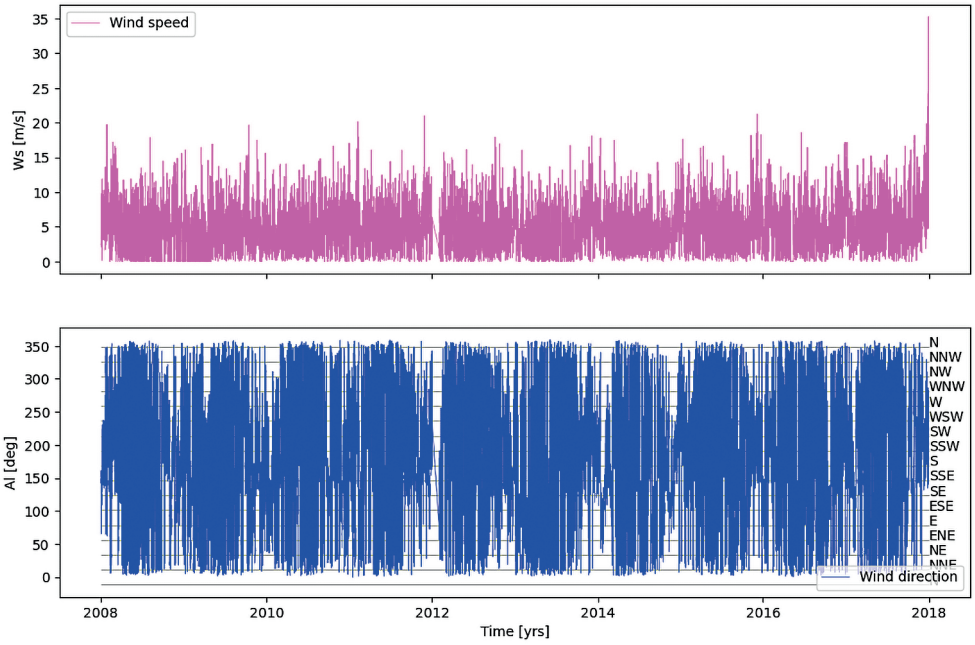

Fig. 2.8. The plot of the wind speed and direction

To generalise the data, the basic statistics of the dataset should be calculated: minimum, maximum, quartiles, mean and standard deviation. The most convenient way to do this is to use the describe() method of the Pandas data frame.

A complete wind data analysis requires a wind speed and direction histogram and an estimation of the parameters of the Weibull distribution applied to the wind speed data. The wind speed histogram can be made using the hist() function from the Matplotlib library (Fig. 2.9).

Wind analysis uses a form of the Weibull distribution other than that available in the Scipy.stats library. Therefore, the most convenient solution would be to use and adapt the rv_continuous object representing the continuous distribution offered by this library. For this purpose, a new distribution object inheriting from rv_continuous is created, containing the required parameter set. The created object represents the Weibull distribution given by the following probability density function formula:

The distribution parameters (,) were determined by the maximum likelihood method. Following this method, a nonlinear equation with respect to the parameter was obtained, which was solved using the newton function from the Scipy.optimize library. The density of the estimated Weibull distribution is shown in Fig. 2.9 with the blue line.

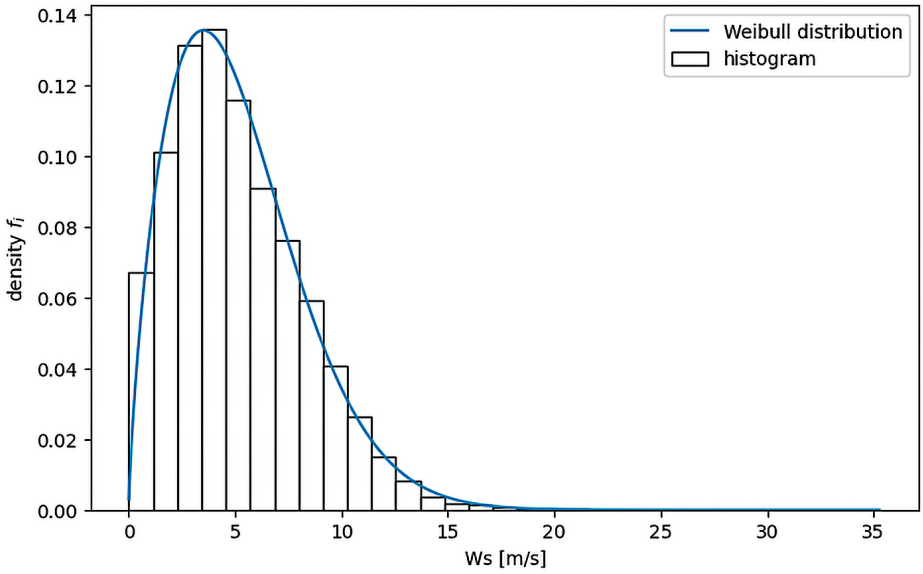

Fig. 2.9. Wind velocity histogram and estimated Weibull distribution density

The last element of wind data analysis is the plot of the so-called wind rose, i.e. a chart that is a histogram of wind speed and direction. Such a plot can be made using the windrose library for Python. The wind rose for the analysed data is presented in Fig. 2.10.

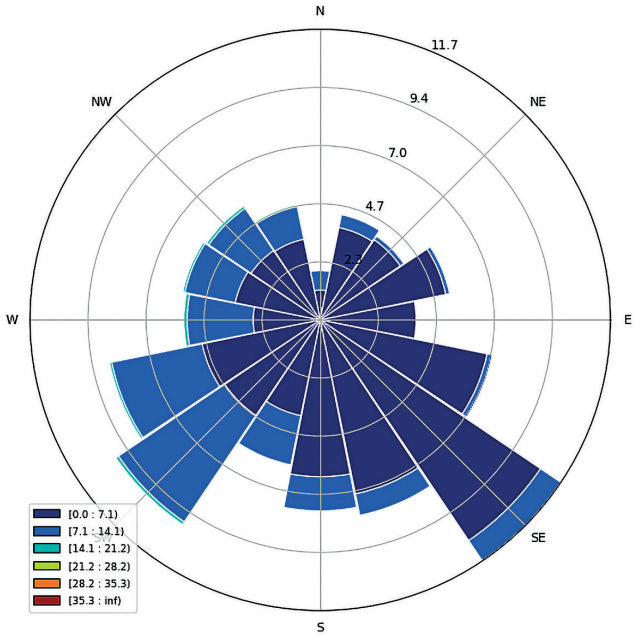

Fig. 2.10. The wind rose, of example data

Final remarks

Data analysis is an important part of the Bridge of Knowledge platform functionality. It is performed using CI TASK compute infrastructure: TRYTON supercomputer (Krawczyk, Nykiel and Proficz, 2015). The provided solution is based on the Jupyter framework deployed within OpenStack (Rosado and Bernardino, 2014) and Kubernetes (Bernstein, 2014) platforms. Jupyter supports researchers as a compute frontend, providing a set of services based on Python and other scripting languages. The framework was widely tested and enriched with a specially developed Stat-reducer package containing useful use case templates.

Future works within the scope of the project and during its maintenance period will cover the following activities: supporting the researchers with new packages dedicated to the framework, maintaining the high performance and scalability of the platforms, providing more use cases and templates.

We believe that Open Research Data analysis supported by the Bridge of Data project will help scientists perform better research work.

References

Bernstein, D. (2014) 'Containers and cloud: From lxc to docker to kubernetes', *IEEE Cloud Computing*, 1(3), pp. 81–84. DOI: 10.1109/MCC.2014.51.

Boettiger, C. (2015) 'An introduction to Docker for reproducible research'. *ACM SIGOPS Operating Systems Review*, 49(1), pp. 71–79. DOI: 10.1145/2723872.2723882.

Brewer, E. A. (2015) 'Kubernetes and the path to cloud native'. *Proceedings of the sixth ACM symposium on cloud computing*, pp. 167–167. DOI: 10.1145/2806777.2809955.

Krawczyk, H., Nykiel, M. and Proficz, J. (2015) 'TRYTON Supercomputer Capabilities for Analysis of Massive Data Streams', *Polish Maritime Research*, 22, 3 (87), pp. 99–104, DOI: 10.1515/pomr-2015-0062.

Kluyver, T. et al. (2016) 'Jupyter Notebooks-a publishing format for reproducible computational workflows', in Loizides, F., Schmidt, B. (ed.) *Positioning and Power in Academic Publishing: Players, Agents and Agendas. ELPUB*, pp. 87-90. DOI: 10.3233/978-1-61499-649-1-87.

Perkel, J.M. (2018) 'Why Jupyter is data scientists' computational notebook of choice', *Nature*, 563 (7732), pp. 145–146. DOI: 10.1038/d41586-018-07196-1.

Ragan-Kelley, M. et al. (2014) 'The Jupyter/IPython architecture: a unified view of computational research, from interactive exploration to communication and publication', *AGU Fall Meeting Abstracts*, pp. H44D-07. Available at: https://agu.confex.com/agu/fm14/meetingapp.cgi/Paper/27181 (Accessed: 8th June 2021).

Rosado, T. and Bernardino, J., (2014) 'An overview of openstack architecture', *Proceedings of the 18th International Database Engineering & Applications Symposium*, pp. 366–367. DOI: 10.1145/2628194.2628195.

Zaharia, M. et al. (2016). Apache spark: a unified engine for big data processing, *Communications of the ACM*, 59(11), pp. 56–65. DOI: 10.1145/2934664.