EPJ Web of Conferences **245**, 05040 (2020) https://doi.org/10.1051/epjconf/202024505040
CHEP 2019

Knowledge sharing on deep learning in physics research using VISPA

Max Beer[1], *Niclas* Eich[1], *Martin* Erdmann[1], *Peter* Fackeldey[1*], *Benjamin* Fischer[1], *Katharina* Hafner[1], *Dennis Daniel Nick* Noll[1], *Yannik Alexander* Rath[1], *Marcel* Rieger[2], *Alexander* Temme[1], *Max* Vieweg[1], and *Martin* Urban[1]

[1]RWTH Aachen University
[2]RWTH Aachen University, now at CERN

Abstract. The VISPA (VISual Physics Analysis) project provides a streamlined work environment for physics analyses and hands-on teaching experiences with a focus on deep learning. VISPA has already been successfully used in HEP analyses and teaching and is now being further developed into an interactive deep learning platform. One specific example is to meet knowledge sharing needs in deep learning by combining paper, code and data at a central place. Additionally the possibility to run it directly from the web browser is a key feature of this development. Any SSH reachable resource can be accessed via the VISPA web interface. This enables a flexible and experiment agnostic computing experience. The user interface is based on JupyterLab and is extended with analysis specific tools, such as a parametric file browser and TensorBoard. Our VISPA instance is backed by extensive GPU resources and a rich software environment. We present the current status of the VISPA project and its upcoming new features.

1 Introduction

The VISPA software [1–6] provides a working environment for scientific analyses in the web browser. Combining a web-browser based implementation with a solid and highly customizable software and feature environment allows scientific research without borders.

The VISPA software serves these functions and even more allows the user to connect to any secure shell (SSH) reachable resource in order to perform a big data physics analysis. By default the user is provided CPU and GPU resources hosted by the VISPA project. JupyterLab [7] is emerging as a similar open-source project. Since June 2019 JupyterLab is available with a first fully stable release as a next-generation web-based user interface for Project Jupyter. It allows the user to work with documents, editors, terminals and custom components while remaining very flexible and extensible. Combining the VISPA project with this high-quality user interface will enhance the user's productivity and interactive experience in the upcoming years for scientific research.

This article is structured as follows. Section 2 outlines the goals of VISPA and discusses the general approach to address these objectives. Section 3 briefly reviews the current implementation. Finally, section 4 introduces the ideas on knowledge sharing on the future VISPA platform.

*presenter, e-mail: peter.fackeldey@rwth-aachen.de

2 Goals and concept

The VISPA project aims to provide a working environment for scientific workflows with a focus on deep learning. The goal is to improve the turnaround of scientific analysis by relieving the user of much non-scientific work. The user's only requirement is a modern web browser. The VISPA project simplifies setting up a sophisticated software- and hardware-environment following four major concepts: connect from anywhere, connect to anywhere, have a quick start and enjoy specialised features. These concepts are discussed in the following four paragraphs.

VISPA is accessible from anywhere and any device, hence a web-based platform suits this goal the best. There is also no need for a software installation. The user can switch between devices conserving personal settings. This allows for continuing the scientific workflow at a later stage or even switching to a different device when the user is abroad.

VISPA allows the user to connect to remote computing systems. These remote systems require only to be SSH reachable and to run a Python interpreter. Figure 1 sketches the connection scheme from client side to the remote resource. The user connects to the VISPA server via HTTP(s) and the server then establishes a secure connection via SSH with the user's credentials to the remote worker node. Using a lightweight remote procedure call (RPC) Python script the communication between the VISPA server and the worker node is settled. The SSH was chosen due to its widespread availability in modern operating system and its security benefits. It encrypts the traffic between the VISPA server and the remote worker node. All sensitive authentication credentials are only forwarded to the resource and never stored.

The VISPA project also aims to provide a default setup for a quick start and thus avoids a steep learning curve. It can be viewed as a software environment with hardware resources. The software environment supports multiple programming languages e.g. C++ and Python. Many libraries of the Python ecosystem are pre-installed with their latest stable release version. A focus is set on machine learning libraries, such as TensorFlow [8], PyTorch [9] and scikit-learn [10]. Additionally the user can install more libraries on their own or request them to be added for all users.
The hardware environment consists of multiple GPU and CPU worker nodes accessed through a HTCondor [11] cluster. The VISPA cluster also provides shared user space across all workers through a network file system (NFS).

A scientific workflow requires specialised tools for analyzing and interpreting data, especially in high energy physics and deep learning research. The VISPA project supports tools to inspect foreign data-formats, such as ROOT, through a JSROOT-based interactive `TBrowser`. The training procedure of deep learning models can be tracked and visualised through a TensorBoard plug-in. Additionally the VISPA project provides a parametric file browser, that enables exploration of high dimensional parameter spaces, which are extracted from arbitrarily nested file paths using regular expressions.

3 The VISPA software

The VISPA software [6] follows the previously outlined schema and consequently consists of three components: client, server, and resource, where the latter is referred to as Workspace.

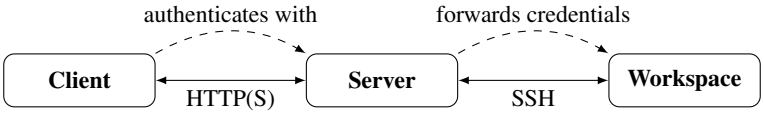

Figure 1: Schema of the three application parts and the client's authentication interactions (dashed).

The graphical interface at the client-side is written in HTML, CSS and JavaScript, while the VISPA server implementation is primarly based on the Python library CherryPy [12]. The communication relies on HTTP(s). User specific data is stored in an external SQL-database accessed by the VISPA server enabling persistent user preferences, even for individual Workspaces. The communication between the VISPA server and the Workspace is enabled by SSH protocol. Using RPyC [13] a very lightweight Python script is running on the Workspace side. These remote procedure calls serve useful aspects such as directory listing with all file informations resulting in reduced latencies. A more detailed overview of the VISPA software can be found in reference [5].

The VISPA software is successfully employed in research, outreach, and education with around 550 users in the last 6 months including undergraduate courses and workshops [14–17].

4 Knowledge sharing with the future VISPA

The future VISPA follows the structure and implementation proposals from reference [5]. Knowledge sharing will be a key part of the future VISPA, which can be realized with three key principles. The combination of the VISPA project and JupyterLab can realize these principles, which will be described in the following.

An interactive work experience, which gives almost immediate feedback of code development, allows users to share knowledge about these developments with respect to a scientific result. This is especially useful for multi-developer projects. Jupyter Notebooks emerged as the leading tool to achieve this. Notebooks can also be shared with others via many different platforms, such as GitHub. The future VISPA spawns a JupyterLab instance on the remote resource and forwards any traffic via the SSH protocol to the VISPA server. Jupyter Notebooks support more than 40 different kernels, allowing the use of any programming language in these Notebooks as long as the language is installed on remote side. The pre-configured software and hardware environment of the VISPA cluster enables new users to explore latest deep learning models seconds after their login.

A very important and direct way of sharing knowledge is to provide minimal code examples with data of scientific findings, such that the user can test the corresponding publication. The VISPA project aims at achieving this in the scope of deep learning in physics research with the future VISPA. A JupyterLab extension is developed within VISPA for this purpose serving as a collection bringing everything together: a reference to the publication, an abstract with the key features and an entrypoint code example with corresponding data. The extension opens in a new tab in the JupyterLab interface, where each element holds a key plot, the title and the abstract of the paper. Additionally there are buttons, linking to the paper itself, to the full code implementation and to a small example, which can be run directly on the VISPA cluster. These code examples together with small datasets originate

from a version control system and can be used on demand on the VISPA cluster.

JupyterLab allows the user to install extensions. There are countless public extensions serving multiple issues, such as LaTeX and version control integration. The VISPA project also comes with several extensions such as the parametric file browser and the above mentioned database for minimal code examples of scientific paper. Additionally every user can install community extensions and develop their own extension enhancing their experience and interaction with their scientific workflow and share those with other users. Each user can perform their own persistent customization of JupyterLab extensions. Such extensions can be searched and installed via the built-in JupyterLab extension manager.

5 Conclusion

The current VISPA project offers users a scientific working environment. It follows a few fundamental concepts: connect from anywhere, connect everywhere, have a quick start and exploit specialised features. Especially the possibility to connect to any SSH reachable resource, which ensures a very flexible working experience, is a key concept of VISPA.
Combining the VISPA software with JupyterLab provides the user with the rich and modern user interface of JupyterLab, and the full flexibility and the quick start offered by VISPA. This allows the future VISPA to take a step towards a knowledge sharing platform. The interactive working experience from Jupyter Notebooks makes it possible for users to share knowledge immediately and work with their colleagues. Additionally the future VISPA is developing a JupyterLab extension, which provides publications and their corresponding minimal code example in one place. In general custom JupyerLab extensions can be develop by any user and shared with others.

Acknowledgments

We wish to thank the conference organizers for their kind support. This VISPA project is supported by the Ministerium für Wissenschaft und Forschung, Nordrhein-Westfalen and the Bundesministerium für Bildung und Forschung (BMBF).

References

[1] H.P. Bretz et al., Journal of Instrumentation **7**, T08005 (2012), 1205.4912
[2] M. Erdmann et al., Journal of Physics: Conference Series **762**, 012008 (2016)
[3] M. Erdmann et al., Journal of Physics: Conference Series **898**, 072045 (2017)
[4] M. Erdmann et al., Journal of Physics: Conference Series **1085**, 042044 (2018)
[5] M. Erdmann et al., EPJ Web Conf. **214**, 05021 (2019)
[6] *VISPA Web* [software], https://git.rwth-aachen.de/3pia/vispa/vispa-web
[7] Project Jupyter, *JupyterLab is Ready for Users*, https://blog.jupyter.org/jupyterlab-is-ready-for-users-5a6f039b8906 (accessed 2020-02-05)
[8] M. Abadi et al., *TensorFlow: Large-scale machine learning on heterogeneous systems* (2015), software available from tensorflow.org, https://www.tensorflow.org/
[9] A. Paszke et al., *Automatic differentiation in PyTorch*, in *NIPS-W* (2017)
[10] F. Pedregosa et al., Journal of Machine Learning Research **12**, 2825 (2011)
[11] D. Thain et al., Concurrency - Practice and Experience **17**, 323 (2005)

[12] *CherryPy* [software], https://cherrypy.org/
[13] *RPyC* [software], https://github.com/tomerfiliba/rpyc
[14] M. Erdmann et al., European Journal of Physics **35**, 035018 (2014)
[15] D. van Asseldonk et al., Journal of Physics: Conference Series **664**, 032031 (2015)
[16] D. van Asseldonk et al., Nuclear and Particle Physics Proceedings **273-275**, 2581 (2016)
[17] *Workshop on Big Data Science in Astroparticle Research*, Aachen, Germany (2017, 2018, 2019), https://indico.scc.kit.edu/event/344,https://indico.scc.kit.edu/event/277,https://indico.scc.kit.edu/event/669/