

A Browser-Based Multi-User Working Environment for Physicists

M. Erdmann, R. Fischer, C. Glaser, D. Klingebiel, M. Komm,
G. Müller, M. Rieger, J. Steggemann, M. Urban, T. Winchen

Physics Institute 3A, RWTH Aachen University, Germany

E-mail: gero.mueller@physik.rwth-aachen.de

Abstract. Many programs in experimental particle physics do not yet have a graphical interface, or demand strong platform and software requirements. With the most recent development of the VISPA project, we provide graphical interfaces to existing software programs and access to multiple computing clusters through standard web browsers. The scalable client-server system allows analyses to be performed in sizable teams, and disburdens the individual physicist from installing and maintaining a software environment. The VISPA graphical interfaces are implemented in HTML, JavaScript and extensions to the Python webserver. The webserver uses SSH and RPC to access user data, code and processes on remote sites. As example applications we present graphical interfaces for steering the reconstruction framework OFFLINE of the Pierre-Auger experiment, and the analysis development toolkit PXL. The browser based VISPA system was field-tested in biweekly homework of a third year physics course by more than 100 students. We discuss the system deployment and the evaluation by the students.

1. Introduction

The growing size and complexity of analysis software and reconstruction frameworks make it increasingly difficult to use and install them on personal computers. At the same time tablet computers and web based applications become more popular. With the VISPA platform, we provide a bridge between these developments. It is a web thin client and web server allowing graphical access to physics software installations on multiple servers using web technologies.

The use cases range from teaching to science. Physics demonstrations can be evaluated by the students during the teaching course. In international collaborations analyses can be created, shared and discussed using the same platform.

In section 2, we give an overview of the system and in section 3 we describe how new features can be added by the means of extensions. In section 4, we report on the application of the VISPA software in a physics course.

2. The VISPA Software

The VISPA platform consists of three tiers: clients, the vispa server and worker nodes. (Fig. 1). This separation enables increased security and scalability. On the client the web browser provides the graphical user interface. The VISPA server is a web server providing the content to the client, which it gathers from the backend servers. Worker nodes can be any computer



ranging from a laptop to a computing cluster. The only requirements for the worker are SSH access and a Python [1] interpreter. Additional software may be required for custom extensions.

2.1. Security

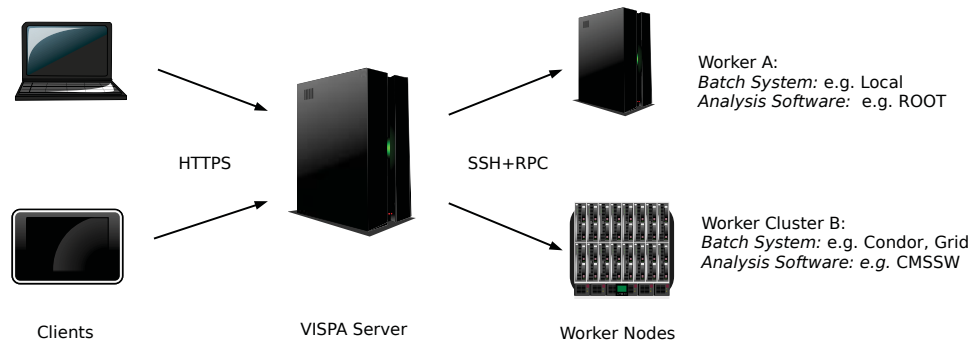


Figure 1. Diagram of the three-tier system. Web thin clients access the VISPA server using a web browser. The VISPA server accesses data and programs on the worker nodes.

Each of three tiers requires its own security considerations. While some can be addressed by the developers, others lie in the responsibility of the deploying administrator.

On the client side, cross-site-scripting is one of the most important issues that needs to be addressed in all extensions. In this threat, malicious JavaScript code uploaded by an attacker could be executed by another user. The communication between the client and the server is encrypted using SSL. The VISPA server addresses, amongst others, SQL injections attempts, secure password hashes and input checks. Remote procedure calls and data exchange between the server and the worker are encrypted using the SSH protocol. On the worker nodes, all data and programs are accessed and executed using dedicated system accounts for each user.

In collaboration with our IT department, a security concept has been developed, which takes the special requirements and demands for public access to the server into account. The security concept includes access control, password protection, privacy, firewall setup, spam prevention, backup, disaster recovery, and logging.

3. Extensions

While it is possible to use existing software via remote shells, graphical interfaces have the advantage of increased usability and productivity. Extensions for common tasks are part of the VISPA platform. File browser, code editor, terminal and job submission are included in VISPA. Screenshots of these extensions are displayed in figures 2, 3 and 6.

Access to batch computing systems is eased by graphical assistance of job management and creation for several batch computing systems. Currently supported are local jobs, HTCondor, LSF and submission to the Grid.

Additionally, specialized extensions to access physics software have been developed. Currently supported are local jobs, as well as submissions to IBM LSF, HTCondor [2], and the Grid using gLite [3].

PXL [4] is a C++ library for high-level physics analyses in high-energy and astroparticle physics. This library implements container classes for physics objects, their relations, a fast file format, a module system and Python bindings. The PXL Analysis designer (Fig. 4) is a graphical interface to create and configure complex analyses using modules for data flow driven analyses. The PXL File Browser (Fig. 5) visualizes the contents of PXL data files.

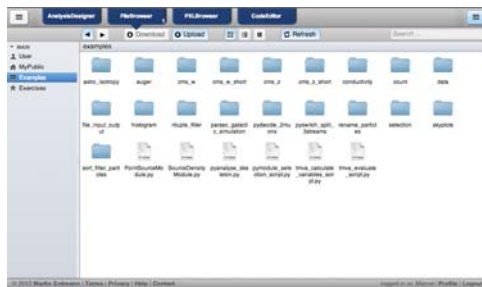


Figure 2. File Manager. Browse and manage files on the worker. Provides common operations and image viewing capabilities.



Figure 3. Code Editor. Edit any file on the worker with syntax highlighting and common shortcuts.

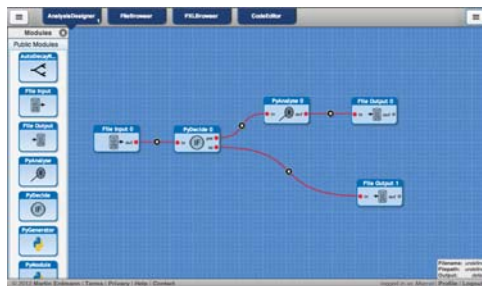


Figure 4. PXL Analysis Designer. PXL features a graphically steered module system, which is the building block for advanced and complex analyses.

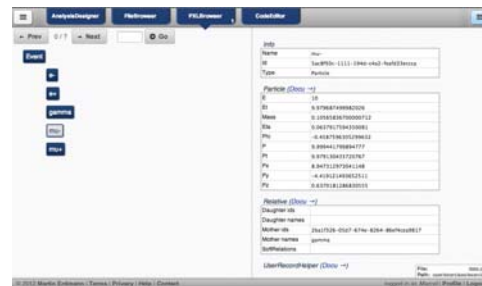


Figure 5. PXL File Browser. Inspect PXL data files, which are optimized for structured HEP events. Attributes and relations of all objects are accessible.

Auger OFFLINE [5] is the modular reconstruction and analysis framework of the Pierre Auger experiment. Information about the available modules and the analyses are stored in complex XML files. The Auger OFFLINE extension included in VISPA (Fig. 7) parses and presents the modules, their options and descriptions in a graphical interface and allows editing the module sequence and module options.

The integration of the extensions in the VISPA platform allows, e.g., the direct submission of jobs created with the PXL or OFFLINE extensions to the batch systems.

Custom extensions can easily be developed and deployed by the administrator of the VISPA server. An extension consists of up to 3 parts, one for each tier. The graphical interface on the client is programmed using HTML, CSS and JavaScript. Python is used to extend the web server and on the worker node. The code for the worker is optional and often just a wrapper around exiting programs. VISPA extensions are bundled as Python packages that can be automatically detected and deployed.

4. Application in Education

The easement of software installation and the graphical interface makes VISPA an apt tool for teaching. It has been deployed at the RWTH Aachen University for a blended-learning initiative in 2012. In the winter term 2012/13, more than 100 3rd year physics students used it to solve data analysis assignments. A course evaluation was conducted and provided valuable feedback on the platform and its application in teaching. The success of the project reflects in

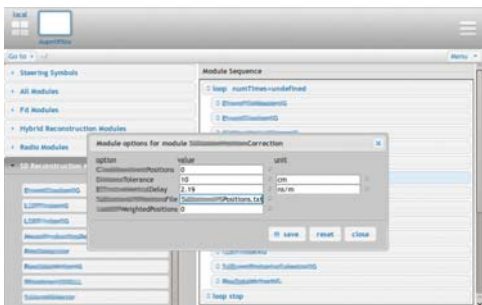


Figure 7. The Auger OFFLINE extension eases the configuration of analyses as all modules and their options, including descriptions, are available in one place.

the following two responses: Many students judged the exercises to deepen their understanding of the course lectures (Fig. 8). The overall assessment of the learning project using the VISPA platform was positive (Fig. 9).

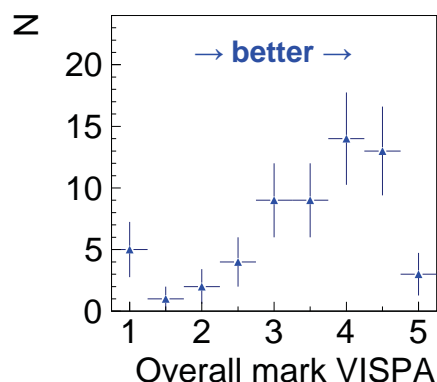


Figure 9. The student’s assessment of the learning project was positive overall. Valuable comment were received on the workflow, e.g. reducing the number of mouse clicks.

5. Conclusion

The VISPA project explores new ideas and ways to develop and execute physics analyses. It provides a platform to develop, execute and share physics analyses and assists in the handling of complex physics software with graphical interfaces.

The three-tiered architecture relieves the users from installing and maintaining software. Students are able to access and share analyses and learn independent of location and time. Scientists are enabled to develop, steer and discuss analyses in international collaborations without technical prerequisites.

References

- [1] van Rossum G Python language website URL <http://www.python.org/>.
- [2] Thain D, Tannenbaum T and Livny M 2005 *Concurrency - Practice and Experience* **17** 323–356
- [3] Cecchi M, Capannini F, Dorigo A, Ghiselli A, Gianelle A *et al.* 2010 *J.Phys.Conf.Ser.* **219** 062039
- [4] Bretz H, Brodski M, Erdmann M, Fischer R, Hinzmann A *et al.* 2012 *JINST* **7** T08005 (*Preprint* 1205.4912)
URL <http://dx.doi.org/10.1088/1748-0221/7/08/T08005>
- [5] Argiro S, Barroso S, Gonzalez J, Nellen L, Paul T C *et al.* 2007
Nucl.Instrum.Meth. **A580** 1485–1496 ISSN 0168-9002 (*Preprint* 0707.1652) URL
<http://www.sciencedirect.com/science/article/pii/S0168900207014106>