VISPA: Direct Access and Execution of Data Analyses for Collaborations

Daniel von Asseldonk, Martin Erdmann, Robert Fischer, Christian Glaser, Gero Müller, Thorben Quast, Marcel Rieger, Martin Urban

III. Physics Institute A, RWTH Aachen University, Aachen, Germany

E-mail: `glaser@physik.rwth-aachen.de`

Abstract. The VISPA project provides a graphical frontend to computing infrastructures. Currently, the focus of the project is to give an online environment for the development of data analyses. Access is provided through a web GUI, which has all functionality needed for working conditions comparable to a personal computer. This includes a new preference system as well as user configurable shortkeys. As all relevant software, data and computing resources are supplied on a common remote infrastructure the VISPA web framework offers a new way of collaborative work where analyses of colleagues can be reviewed and executed with just one click. Furthermore, VISPA can be extended to the specific needs of an experiment or other scientific use cases. This is presented in the form of a new GUI to the analysis framework Offline of the Pierre Auger collaboration.

1. Introduction

We are facing the situation that on the one hand our mobile computing devices become smaller and more portable whereas - on the other hand - more computing resources are needed and larger and larger data volumes need to be handled. Furthermore, it is getting more difficult to install software needed for data analyses on an ultrabook or tablet, even disregarding computing power and storage.

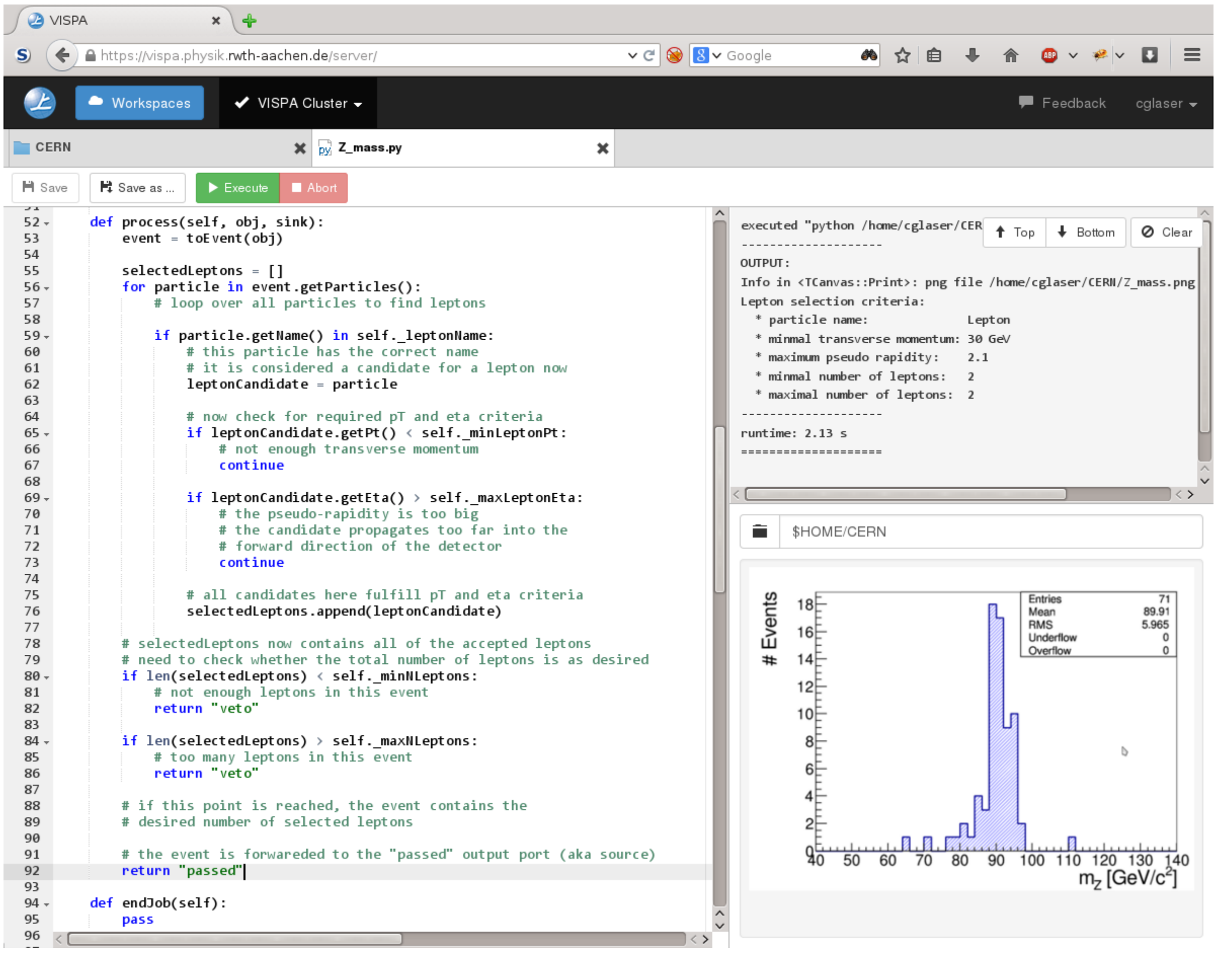
One solution is to use a software that all devices have in common: The web browser. Instead of installing software and transferring data onto a device, one opens a browser, connects to a web server and accesses a working environment by means of a web application.

The VISPA software provides this functionality. VISPA is a graphical front-end to infrastructures which makes software, data and computing resources available through the web.

The article is structured as follows: We first describe the software and its technical implementation in detail. After describing how VISPA can be extended by custom applications, current and future use cases are discussed.

2. The software

The VISPA software [1, 2] consists of two parts: The first part is the basic functionality which provides a GUI framework, user management and the communication between client (web browser), server and worker nodes. The second part are applications that use the VISPA framework. Several applications are shipped directly with the main VISPA package. These applications are a file browser, a terminal emulator and a code editor. However, the concept of



```python
    def process(self, obj, sink):
        event = toEvent(obj)

        selectedLeptons = []
        for particle in event.getParticles():
            # loop over all particles to find leptons

            if particle.getName() in self._leptonName:
                # this particle has the correct name
                # it is considered a candidate for a lepton now
                leptonCandidate = particle

                # now check for required pT and eta criteria
                if leptonCandidate.getPt() < self._minLeptonPt:
                    # not enough transverse momentum
                    continue

                if leptonCandidate.getEta() > self._maxLeptonEta:
                    # the pseudo-rapidity is too big
                    # the candidate propagates too far into the
                    # forward direction of the detector
                    continue

                # all candidates here fulfill pT and eta criteria
                selectedLeptons.append(leptonCandidate)

        # selectedLeptons now contains all of the accepted leptons
        # need to check whether the total number of leptons is as desired
        if len(selectedLeptons) < self._minNLeptons:
            # not enough leptons in this event
            return "veto"

        if len(selectedLeptons) > self._maxNLeptons:
            # too many leptons in this event
            return "veto"

        # if this point is reached, the event contains the
        # desired number of selected leptons

        # the event is forwareded to the "passed" output port (aka source)
        return "passed"

    def endJob(self):
        pass
```

```
executed "python /home/cglaser/CER
--------------------
OUTPUT:
Info in <TCanvas::Print>: png file /home/cglaser/CERN/Z_mass.png
Lepton selection criteria:
  * particle name:              Lepton
  * minmal transverse momentum: 30 GeV
  * maximum pseudo rapidity:    2.1
  * minmal number of leptons:   2
  * maximal number of leptons:  2
--------------------
runtime: 2.13 s
====================
```

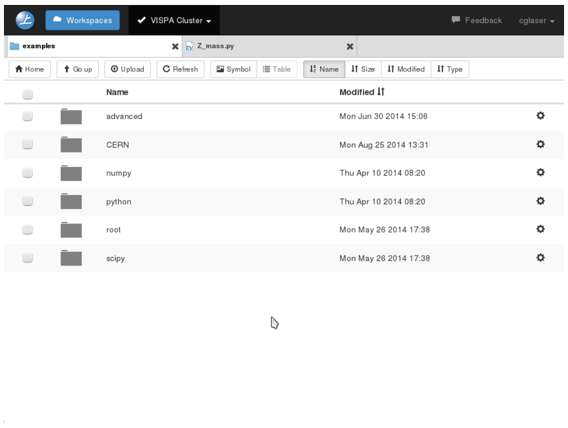
Figure 1. Screenshot of the code editor application.

VISPA is to provide a web framework that can then be extended with custom made applications by other groups themselves. This is demonstrated in section 4 by means of a graphical steering of the analysis framework of the Pierre Auger collaboration.

In the current version 1.0 of the VISPA platform, the GUI has been redesigned to optimize the workflow providing working conditions comparable to a local desktop environment. A preference system is used to satisfy users' individual tastes of the GUI appearance and working style. Frequently used operations are accessible with just one mouse click, or by using a shortkey respectively. Shortcuts are user configurable on the whole platform.

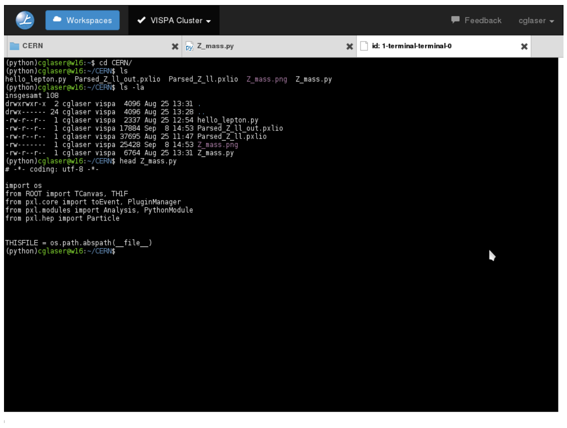
Figures 1, 2a and 2b show screenshots of the three basic VISPA applications. The main GUI elements are visible in the upper menu bar. A VISPA application is opened as a tab and multiple applications can be opened at the same time. The code editor application was extended to instantly execute python scripts. With a click on the green execute button the script is executed. The terminal output of the python job is visible in the upper right part of the application window. Output plots that are generated by the job are visible in the lower right part. A click on a plot will open a larger preview.

The VISPA platform consists of three tiers: Clients, the VISPA server and worker nodes (figure 3a). This separation enables increased scalability [3]. On the client side 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



Name	Modified
advanced	Mon Jun 30 2014 15:06
CERN	Mon Aug 25 2014 13:31
numpy	Thu Apr 10 2014 08:20
python	Thu Apr 10 2014 08:20
root	Mon May 26 2014 17:38
scipy	Mon May 26 2014 17:38

(a)



```
(python)cglaser@wl6:~$ cd CERN/
(python)cglaser@wl6:~/CERN$ ls
hello_lepton.py  Parsed_Z_ll_out.pxlio  Parsed_Z_ll.pxlio  Z_mass.png  Z_mass.py
(python)cglaser@wl6:~/CERN$ ls -la
insgesamt 108
drwxrwxr-x  2 cglaser vispa  4096 Aug 25 13:31 .
drwx------ 24 cglaser vispa  4096 Aug 25 13:28 ..
-rw-r--r--  1 cglaser vispa  2337 Aug 25 12:54 hello_lepton.py
-rw-r--r--  1 cglaser vispa 17884 Sep  8 14:53 Parsed_Z_ll_out.pxlio
-rw-r--r--  1 cglaser vispa 37695 Aug 25 11:47 Parsed_Z_ll.pxlio
-rw-------  1 cglaser vispa 25428 Sep  8 14:53 Z_mass.png
-rw-r--r--  1 cglaser vispa  6764 Aug 25 13:31 Z_mass.py
(python)cglaser@wl6:~/CERN$ head Z_mass.py
# -*- coding: utf-8 -*-

import os
from ROOT import TCanvas, TH1F
from pxl.core import toEvent, PluginManager
from pxl.modules import Analysis, PythonModule
from pxl.hep import Particle


THISFILE = os.path.abspath(__file__)
(python)cglaser@wl6:~/CERN$
```

(b)

Figure 2. Screenshot of the file browser (a) and the terminal (b) application.

ranging from a laptop to a computing cluster. The only requirements for the workers are SSH access and a Python [4] interpreter. Additional software may be required for custom VISPA applications.

The GUI (client code) is based on the HTML5 markup language, CSS3 and the JavaScript scripting language, which are supported by common web browsers [5]. Advanced functionality is implemented using the JavsSript library jQuery [6] and the Bootstrap3 framework [7]. Furthermore, most web content is generated by rendering HTML templates on server or client side using mako [8] or transparency [9].

Server and workspace code is written in Python [4]. The server is build on the web framework CherryPy [10]. This allows for a convenient integration of various scientific software packages written in Python, such as PXL [2], ROOT [11] and SciPy [12]. Persistent data, e.g. user management and preferences, are stored in databases using SQLAlachemy [13].

Server - workspace communication The communication between server and worker node (workspace) is established using a bootstrap approach: At the beginning of the communication, all relevant software is copied via ssh into the memory of the worker node. This makes a SSH access and a Python [4] interpreter the only requirement.

The first step is to create a ssh connection and execute a minimal Python code which makes the stdin and stdout pipe to exchange commands. Then all required Python packages are compressed into one zip archive on the server and copied and extracted into the workers memory. All further communication uses remote procedure calls. In our implementation we make use of the Python packages RPyC [14] and paramiko [15].

3. Extending VISPA with custom applications

VISPA is designed to be extended to custom needs of a user or a group. The VISPA framework provides all necessary functionality to utilize the client - server - worker system. Hence, the user needs to implement only the specific tasks of his application.

The VISPA source code [16] contains two well documented example applications, the so-called ”dummy” and ”demo” application. They demonstrate the implementation of a VISPA app and show most features of the VISPA framework: Utility functions are provided which simplify the server - client communication using Ajax, as well as the server - workspace communication. Furthermore, intercommunication between different apps is supported. For instance a file

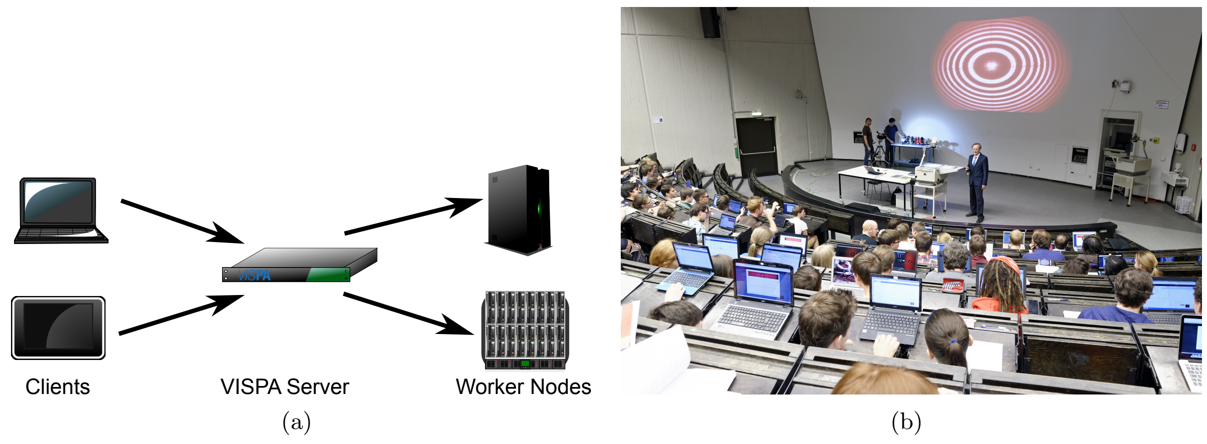


Figure 3. (a) Server setup. (b) Usage of VISPA in an experimental physics lecture. Picture: Peter Winandy (Aachen).

selection dialog can be opened as known from desktop applications.

4. Graphical steering of Pierre Auger analysis framework

An application which demonstrates the extensibility of the VISPA platform is the Auger Offline application which provides a graphical steering of the analysis framework Offline [17, 18] of the Pierre Auger collaboration. The reconstruction and analysis of raw data is steered by defining a sequence of modules and by optionally overriding the default module options. This is normally achieved by creating XML steering files which require knowledge of all necessary modules and module options including their correct spelling.

To simplify the usage of Offline, the VISPA framework was used to implement a GUI which can be seen in figure 4. In the left panel all available modules are displayed grouped by categories. In the middle panel, the module sequence can be defined. Via drag and drop modules can be moved from the left to the middle panel or sorted within the module sequence.

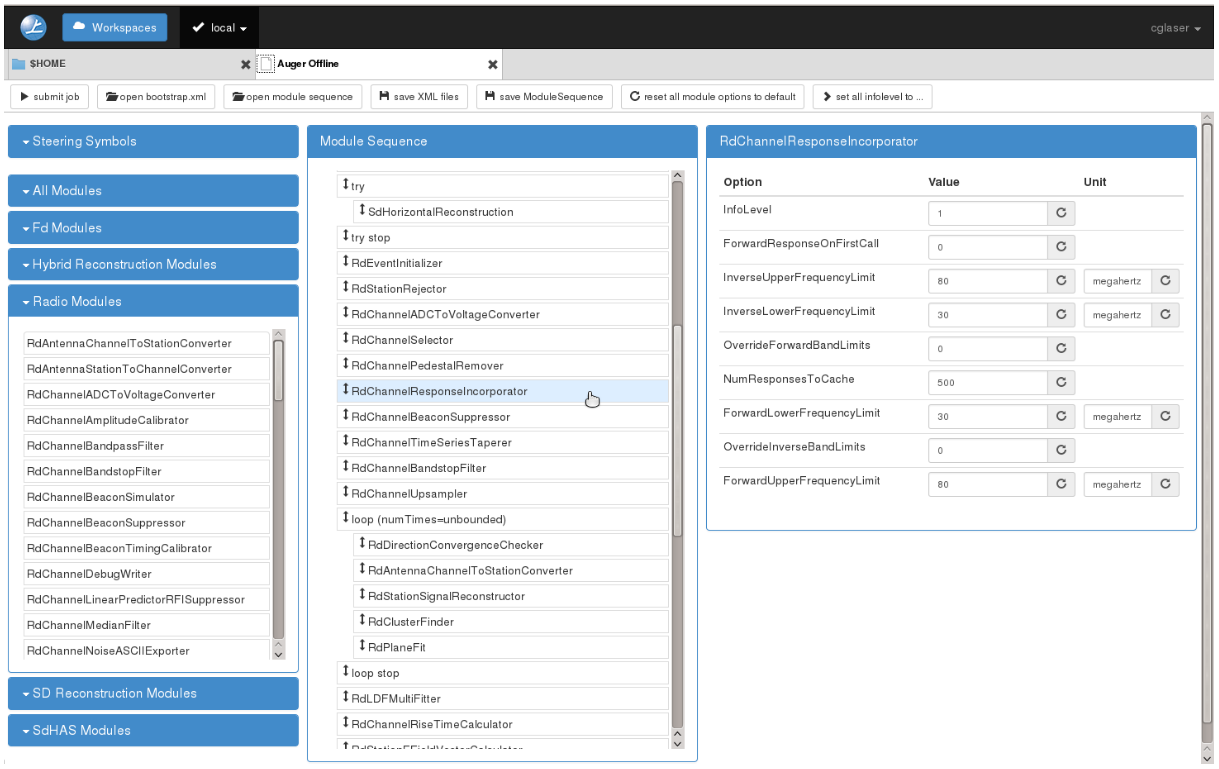
A click on one module opens the right panel where the module's options can be edited. To further assist the user, a tool-tip explaining the meaning of the option is displayed when hovering over an option. Furthermore, the user can take advantage of VISPA's job-management apps. With a few clicks one or multiple jobs can be created and submitted to different batch systems.

This VISPA application was written without modification of the Auger analysis framework Offline. All information is obtained dynamically by parsing XML configuration files that come with the Offline software. Hence, the VISPA app does not interfere with the development of Offline and will always be up to date to the installed version of the workspace.

The main advantages of using VISPA instead of a desktop GUI framework are: The application is usable from any device and one does not need to run the application on the computer where Offline is installed. The GUI is always up to date to the installed version on the computer or computing cluster that is accessed through VISPA and one can work directly on the remote files or submit a job. The device which steers Offline is decoupled from the device providing the computing resources.

5. Collaborative analyses

VISPA opens a new way of collaborative work. As the analysis as well as all relevant software and data are supplied on a common remote infrastructure, an analysis can be shared with colleagues by giving them access to the relevant files. The analysis can be reviewed or executed through


Workspaces
local
cglaser
$HOME
Auger Offline
submit job
open bootstrap.xml
open module sequence
save XML files
save ModuleSequence
reset all module options to default
set all infolevel to ...
Steering Symbols
All Modules
Fd Modules
Hybrid Reconstruction Modules
Radio Modules
RdAntennaChannelToStationConverter
RdAntennaStationToChannelConverter
RdChannelADCToVoltageConverter
RdChannelAmplitudeCalibrator
RdChannelBandpassFilter
RdChannelBandstopFilter
RdChannelBeaconSimulator
RdChannelBeaconSuppressor
RdChannelBeaconTimingCalibrator
RdChannelDebugWriter
RdChannelLinearPredictorRFISuppressor
RdChannelMedianFilter
RdChannelNoiseASCIIExporter
SD Reconstruction Modules
SdHAS Modules
Module Sequence
try
SdHorizontalReconstruction
try stop
RdEventInitializer
RdStationRejector
RdChannelADCToVoltageConverter
RdChannelSelector
RdChannelPedestalRemover
RdChannelResponseIncorporator
RdChannelBeaconSuppressor
RdChannelTimeSeriesTaperer
RdChannelBandstopFilter
RdChannelUpsampler
loop (numTimes=unbounded)
RdDirectionConvergenceChecker
RdAntennaChannelToStationConverter
RdStationSignalReconstructor
RdClusterFinder
RdPlaneFit
loop stop
RdLDFMultiFitter
RdChannelRiseTimeCalculator
RdChannelResponseIncorporator

Option	Value	Unit
InfoLevel	1	
ForwardResponseOnFirstCall	0	
InverseUpperFrequencyLimit	80	megahertz
InverseLowerFrequencyLimit	30	megahertz
OverrideForwardBandLimits	0	
NumResponsesToCache	500	
ForwardLowerFrequencyLimit	30	megahertz
OverrideInverseBandLimits	0	
ForwardUpperFrequencyLimit	80	megahertz

Figure 4. Screenshots of the VISPA application to steer the analysis framework Offline of the Pierre Auger collaboration.

the VISPA GUI giving a significant improve in comfort compared to remote ssh access. It is planned to add GUI interfaces to create working groups and to share analyses with other VISPA users as well as a GUI interface to common repository systems to facilitate shared work.

6. Application in education and outreach

Having a web browser as the only requirement for the user, VISPA is ideal to be used in education and outreach activities.

VISPA has successfully been used to supplement a 2nd and 3rd years experimental physics course at RWTH Aachen University in a blended learning initiative [19, 20]: Students analyzed lecture hall experiments themselves directly during the lecture (see figure 3b). After the experiment has been performed the data are uploaded on the VISPA server. A python analysis script is provided which should be extended by the students. Thereby the amount of what the students need to implement by themselves is adapted to the skills of the course. By using their own laptops, the students can log in to the VISPA server and analyze the measured data. An often task e.g. is to fit an analytic function to the data points and thereby determining fundamental physical constants such as the g-factor of an electron. More advanced analysis tasks are given as homework which are discussed in a weekly tutorial.

Furthermore, VISPA is used for CERN outreach activities for its 60th anniversary to perform a data analysis on public CMS data [21]. For example, the mass of the Z-boson can be reconstructed.

7. Conclusions

The VISPA web framework enables data analyses in a web browser. It provides a platform to develop, execute and share physics analyses. With the basic applications that are shipped with the VISPA package, all tools needed to develop data analyses are provided. As all relevant software, data and computing resources can be provided on a remote infrastructure that can be accessed through the VISPA GUI, analyses can be shared with colleagues keeping the comfortable graphical interface.

Furthermore, VISPA is designed to be extended with custom applications to satisfy the needs of specific user groups and unforeseen use cases. All necessary general functionality to write a web application is provided so that the user can focus on the specific task of his application.

Achnowledgments

We wish to thank the organizers of the ACAT2014 conference for their kind support. This work is supported by the Ministerium für Wissenschaft und Forschung, Nordrhein-Westfalen, the Bundesministerium für Bildung und Forschung (BMBF) and the Deutsche Forschungsgemeinschaft (DFG).

References

[1] VISPA web page `http://vispa.physik.rwth-aachen.de`
[2] Bretz H-P et al 2012 *J. Instrum.* **7** T08005 doi: 10.1088/1748-0221/7/08/T08005
[3] Erdmann M et al 2014 *J. Phys. Conf. Ser.* **513** 62034 doi:10.1088/1742-6596/513/6/062034
[4] van Rossum G Python language `http://www.python.org/`.
[5] Erdmann M et al 2014 *J. Phys. Conf. Ser.* **523** 012021 doi: 10.1088/1742-6596/523/1/012021
[6] jQuery `http://jquery.com/`
[7] Bootstrap `http://getbootstrap.com/`
[8] Mako `http://www.makotemplates.org/`
[9] Transparency `http://leonidas.github.io/transparency/`
[10] A Minimalist Python Web Framework `http://www.cherrypy.org`
[11] Antcheva I et al 2009 *Comput. Phys. Commun.* **180** 2499 doi: 10.1016/j.cpc.2009.08.005
[12] SciPy `http://www.scipy.org`
[13] SQL alchemy `http://www.sqlalchemy.org/`
[14] RPyC `http://rpyc.readthedocs.org/`
[15] Paramiko `http://www.paramiko.org/`
[16] VISPA source code `https://forge.physik.rwth-aachen.de/projects/vispa-web`
[17] Argirò S et al 2007 *Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip.* **580** 1485 doi: 10.1016/j.nima.2007.07.010
[18] Abreu P et al 2011 *Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip.* **635** 92 doi: 10.1016/j.nima.2011.01.049
[19] Bender H, Sauter A and Sauter W 2002 *Blended Learning: Effiziente Integration von E-Learning und Präsenztraining* Neuwied: Luchterhand.
[20] Erdmann M et al 2014 *Eur.J.Phys.* **35** 35018 doi: 10.1088/0143-0807/35/3/035018
[21] CERN outreach `http://opendata.cern.ch/`