Data Mining as a Service (DMaaS)

E. Tejedor*, D. Piparo*, L. Mascetti*, J. Moscicki*, M. Lamanna*, P. Mato*

*** CERN, CH-1211 Geneva 23, Switzerland**

Abstract. Data Mining as a Service (DMaaS) is a software and computing infrastructure that allows interactive mining of scientific data in the cloud. It allows users to run advanced data analyses by leveraging the widely adopted Jupyter notebook interface. Furthermore, the system makes it easier to share results and scientific code, access scientific software, produce tutorials and demonstrations as well as preserve the analyses of scientists.

This paper describes how a first pilot of the DMaaS service is being deployed at CERN, starting from the notebook interface that has been fully integrated with the ROOT analysis framework, in order to provide all the tools for scientists to run their analyses. Additionally, we characterise the service backend, which combines a set of IT services such as user authentication, virtual computing infrastructure, mass storage, file synchronisation, development portals or batch systems. The added value acquired by the combination of the aforementioned categories of services is discussed, focusing on the opportunities offered by the CERNBox synchronisation service and its massive storage backend, EOS.

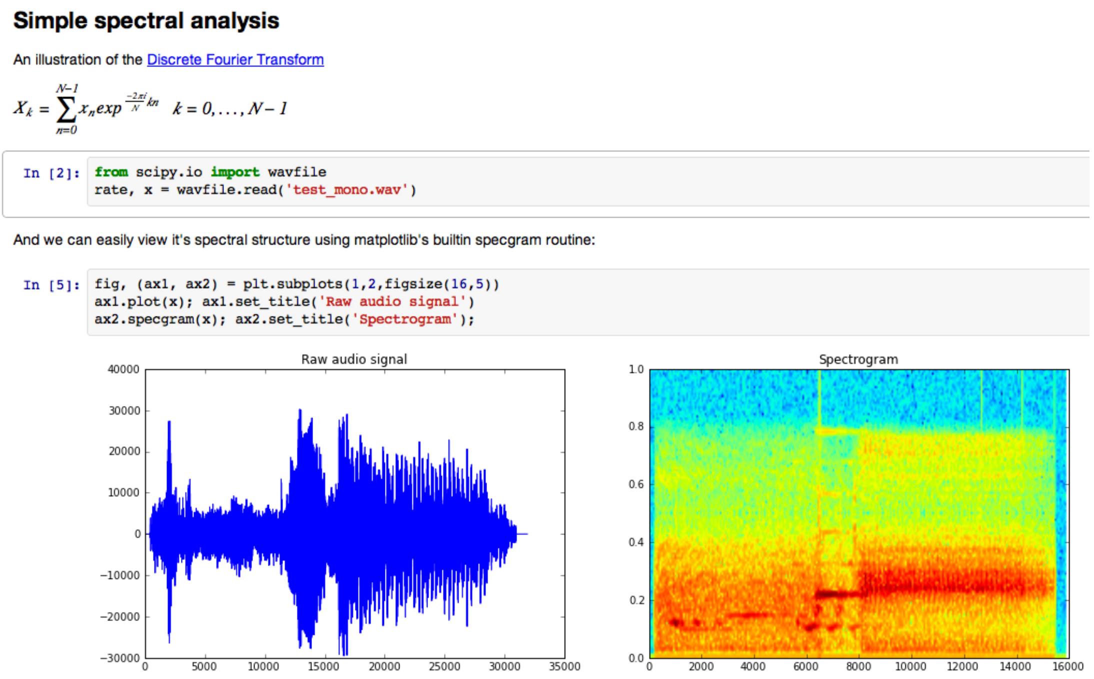
1. Introduction

Not only High-Energy Physics is confronted with challenging data processing and analysis needs, often referred to as "big data". Numerous communities are innovating in this field by contributing ideas for alternative analysis workflows and new tools. Among the directions explored, there is a noticeable trend towards web-based interactive analysis backed up by data and computing resources accessed via "the cloud" [1, 2, 3]. Such trends are coupled with the provision of software as a service, which allows users to focus on the solution for the problem to be solved rather than on installation, configuration and operational matters. These circumstances have suggested to us a rethink of our data analysis model, extending the capabilities of our tools to complement the existing techniques in order to allow even more effective approaches in a shared environment leveraging the available IT infrastructures in an optimal way.

In that sense, this paper presents the Data Mining as a Service (DMaaS) project, which aims to provide a software (see section 2) and distributed computing infrastructure (see section 3) for interactive analysis of scientific data in the cloud. DMaaS eases the sharing of results and scientific code, the access to scientific software, the creation of tutorials and demonstrations as well as the preservation the analyses of scientists.

2. The Notebooks: One of the Interfaces of DMaaS

Nowadays, a common approach for interactive data analysis is to combine code, text, plots and rich media in the same document, known as "notebook". Notebooks are usually divided



Simple spectral analysis

An illustration of the Discrete Fourier Transform

$$X_k = \sum_{n=0}^{N-1} x_n exp^{\frac{-2\pi i}{N} kn} \quad k = 0, \ldots, N-1$$

```
In [2]: from scipy.io import wavfile
        rate, x = wavfile.read('test_mono.wav')
```

And we can easily view it's spectral structure using matplotlib's builtin specgram routine:

```
In [5]: fig, (ax1, ax2) = plt.subplots(1,2,figsize(16,5))
        ax1.plot(x); ax1.set_title('Raw audio signal')
        ax2.specgram(x); ax2.set_title('Spectrogram');
```

Figure 1. Example of a Jupyter notebook.

in cells where the user can type code, execute it and see the results inline. A particularly successful notebook environment is the one provided by Jupyter [4], which can be thought of as an interactive programming shell running in a web browser. It is an agile tool for both exploratory computation and data mining, and provides a platform to support reproducible research, since all inputs and outputs may be stored in a one-to-one way in notebooks. Figure 1 depicts an example of a Jupyter notebook document where markdown text, formulae, code and plots are combined.

The aforementioned attractive features of the Jupyter notebooks motivated their choice as the main interface of the DMaaS service.

2.1. Integration of ROOT and Notebooks

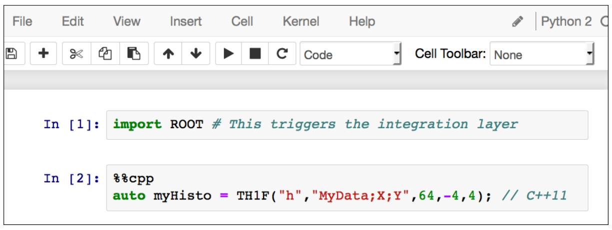
Jupyter is not restricted to a particular language, but instead it allows to plug in language extensions known as *kernels*. It can also accommodate various ecosystems of tools for data analysis, e.g. R [5] or pandas [6].

In order to make ROOT [7] a first-class citizen of Jupyter too, we developed a software layer to integrate both technologies. As a result, users can write *ROOTbooks* (i.e. ROOT notebooks), characterised by the following features:

(i) *Two languages of choice*: a new Jupyter kernel for ROOT was developed and, as a result, ROOT-based analyses can be written in C++. Additionally, the ROOT Python interface, PyROOT, was enhanced to fully exploit its usage in notebooks.

(ii) *Language interoperability*: it is possible to mix C++ and Python in the same ROOTbook, as well as to use C++ variables/functions/classes from Python and vice versa.

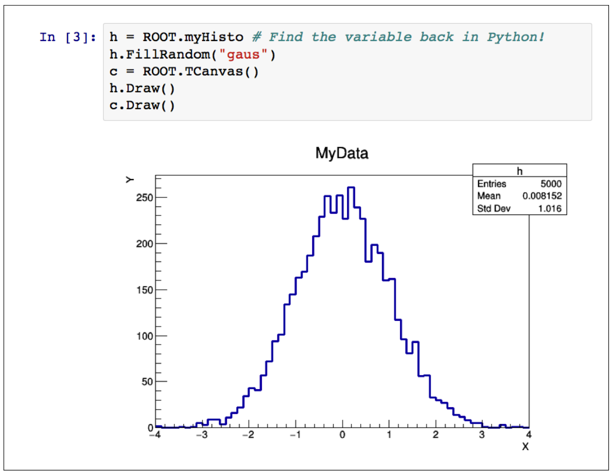
(iii) *Static and interactive data visualisation*: graphics can be displayed in the notebook. ROOTbooks support two modes for graphics: static and JavaScript based; while the former



```
In [1]: import ROOT # This triggers the integration layer

In [2]: %%cpp
        auto myHisto = TH1F("h","MyData;X;Y",64,-4,4); // C++11
```

(a)



```
In [3]: h = ROOT.myHisto # Find the variable back in Python!
        h.FillRandom("gaus")
        c = ROOT.TCanvas()
        h.Draw()
        c.Draw()
```

(b)

Figure 2. A Python ROOTbook.

embeds an image in the document, the latter shows interactive plots that can be modified (e.g. by zooming) and saved as an image.

(iv) *Tab completion*: equivalent to that of the ROOT prompt.

2.2. Programming Model Example

In order to illustrate the features enumerated in section 2.1, this section shows an example of how to write a ROOTbook in Python and C++.

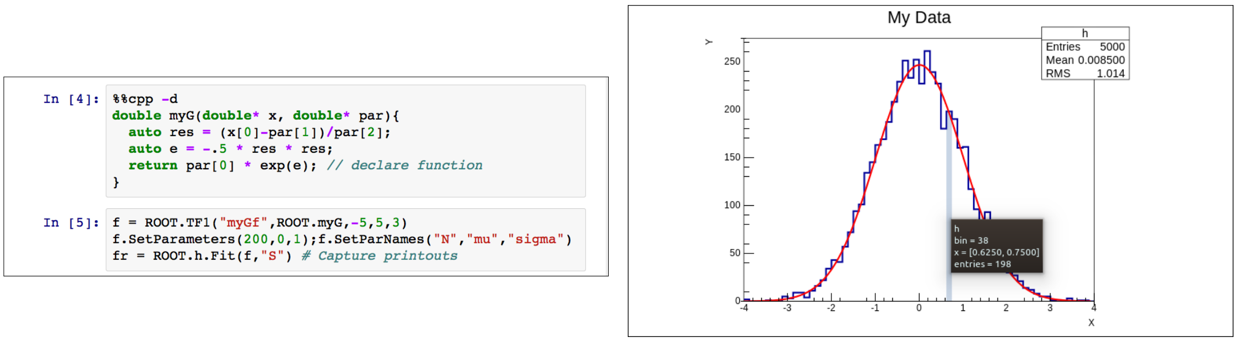
In Python, the user starts by importing the ROOT Python module so that all the ROOTbook features are activated, as can be seen in the first cell of Figure 2(a). After that, she can choose to either continue writing in Python or to switch to C++ in the next cell; in the latter scenario, the **%%cpp** magic is used: in Jupyter, magics can modify the behaviour of a particular cell, in this case transforming it into a C++ cell that creates a histogram (still in Figure 2(a)). It is worth noting how this C++ cell is highlighted in C++, differently from the first (Python) cell.

The third cell (Figure 2(b)) is an example of Python-C++ interoperability, since the histogram created in C++ is accessed from Python, filled and drawn. As a result of the drawing, the histogram is displayed inline in the notebook.

Special mention is deserved by the interactive graphics of the ROOTbooks, implemented with JSROOT [8]. Continuing with the example, the code in Figure 3(a) defines a function in C++ and uses that function in Python to fit a histogram. After that, if the JavaScript visualisation mode is enabled and the canvas is drawn, the embedded display that appears corresponds to a JavaScript object, with which the user can interact by e.g. inspecting the content of the histogram bins (Figure 3(b)).

Finally, regarding C++ ROOTbooks, they also benefit from all the functionalities described so far, the only difference being the language used.

The integration of ROOT and notebooks reached production quality. These new features were already used to document ROOT features in a modern way, e.g. on the framework's website (see [9]) and to provide an online ROOT demo on Binder [10] (see [11]).



```
In [4]: %%cpp -d
        double myG(double* x, double* par){
          auto res = (x[0]-par[1])/par[2];
          auto e = -.5 * res * res;
          return par[0] * exp(e); // declare function
        }

In [5]: f = ROOT.TF1("myGf",ROOT.myG,-5,5,3)
        f.SetParameters(200,0,1);f.SetParNames("N","mu","sigma")
        fr = ROOT.h.Fit(f,"S") # Capture printouts
```

(a) (b)

Figure 3. JavaScript visualisation in a ROOTbook.

3. DMaaS and the Portfolio of CERN Services

Section 2 described the Jupyter notebook interface and its integration with ROOT. The main objective of the DMaaS project is to leverage that interface to provide a service of notebooks (and ROOTbooks) on demand.

In order to develop the DMaaS service, CERN will rely on existing, production-grade technologies such as:

- CVMFS [12] for the distribution of the software in the cloud, including ROOT and LHC experiments' software stacks.
- OpenStack [13] for the provision of the CPU resources necessary to run the users' notebooks and their calculations.
- EOS [14] as mass storage, hosting for example experimental and simulation data.
- CERNBox [15], the CERN file synchronisation and share service, for storing user files.

Figure 4 shows the DMaaS deployment on top of the CERN services' portfolio. The login happens with CERN credentials and the orchestration of users is delegated to JupyterHub [16]. The execution of notebooks is encapsulated in Docker containers [17], scheduled on the OpenStack infrastructure. The individual containers are lightweight: the needed software (e.g. ROOT) is available on CVMFS. Access to the CERN mass storage EOS is available, and the user's home directory in each container is the EOS portion private to each user synchronised via CERNBox. The synchronisation of local directories is one of the gateways to access the cloud.

With such a service and by means of only a web browser, users will be able to write and execute data analyses, see their results inlined (text, graphics) and combine those with explanations about what they are doing, everything in the same document. When finished, notebooks could be saved and shared with other colleagues who could review, modify and re-run those notebooks.

4. Conclusions and Future Work

In the context of the DMaaS project, the ROOT and Jupyter technologies have been successfully integrated: C++ and Python interfaces are supported in production. Documentation, tutorials and online demos showing advanced features like interactive JavaScript graphics have been created.

Within the framework of a coherent view at CERN, a cloud-based data analysis model is being made available to scientists. Its implementation consists in the JupyterHub deployment of DMaaS, which relies entirely on production-grade services offered by the CERN IT department.

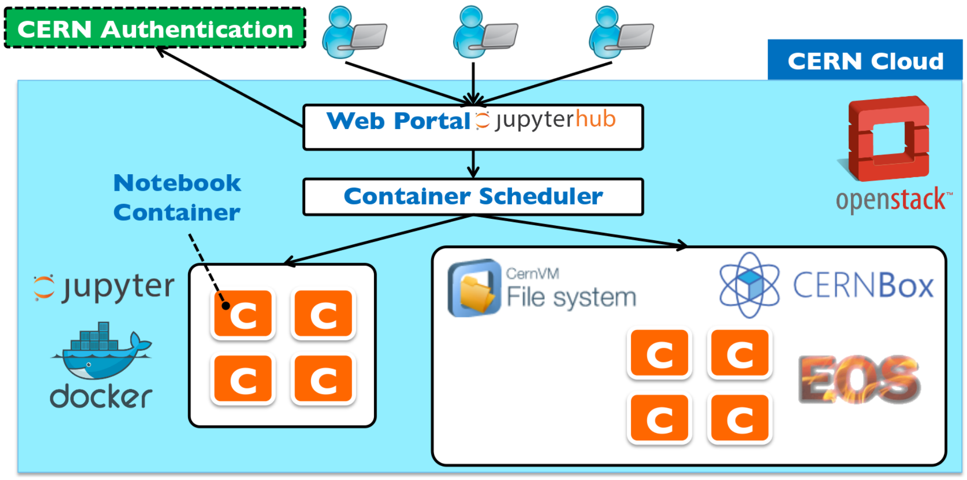


Figure 4. The Jupyter deployment model of DMaaS.

In the future, the current design of DMaaS will be generalised so that it can be integrated with other ecosystems of technologies (different from CERN's). Additionally, a tighter integration of ROOT and other well-established data analysis ecosystems such as pandas will be investigated in the context of the Jupyter notebooks. Finally, the possibility of on-demand exploitation of clusters for distributing calculations spawned from within notebooks will be assessed.

References

[1] Wakari: Web-based Python Data Analysis. `http://wakari.io/`
[2] SageMathCloud: Collaborative Computational Mathematics. `http://cloud.sagemath.com/`
[3] Plotly: Make charts and dashboards online. `http://plot.ly/`
[4] Jupyter: Open source, interactive data science and scientific computing. `http://jupyter.org/`
[5] The R Project for Statistical Computing. `http://www.r-project.org/`
[6] Python Data Analysis Library. `http://pandas.pydata.org/`
[7] R. Brun et al. 1997 ROOT - An Object Oriented Data Analysis Framework *Nucl. Inst. Meth. In Phys.* **A 389** pp 81-86
[8] B Bellenot et al.; 2015 J. Phys.: Conf. Ser. 664 062033,“JavaScript ROOT”
[9] ROOT website, code examples. `http://root.cern.ch/code-examples`
[10] Binder. `http://mybinder.org`
[11] ROOT Binder repository. `http://root.cern.ch/rootbinder`
[12] J Blomer et al.; 2011 J. Phys.: Conf. Ser. 331 042003, “Distributing LHC application software and conditions databases using the CernVM file system”
[13] OpenStack: Open Source Cloud Computing Software. `http://www.openstack.org`)
[14] A Peters et al.; 2015 J. Phys.: Conf. Ser. 664 042042, “EOS as the present and future solution for data storage at CERN”
[15] L Mascetti et al.; 2015 J. Phys.: Conf. Ser. 664 062037, “CERNBox + EOS: end-user storage for science”
[16] JupyterHub: A multi-user server for Jupyter notebooks. `https://github.com/jupyter/jupyterhub`
[17] Docker: build, ship and run any app, anywhere. `https://www.docker.com`