An Analysis Framework for KCDC

Frank Polgart[1 [0000−0002−9324−7146]], Andreas Haungs[1 [0000−0002−9638−7574]], Donghwa Kang[1 [0000−0002−5149−9767]], Doris Wochele[1 [0000−0001−6121−0632]], Jürgen Wochele[1 [0000−0003−3854−4890]], and Victoria Tokareva[1 [0000−0001−6699−830X]]

Karlsruhe Institute of Technology, Institute for Nuclear Physics, 76021 Karlsruhe, Germany
frank.polgart@kit.edu

Abstract. We will introduce a data analysis extension for the KASCADE Cosmic-ray Data Center (KCDC), based on the Jupyterhub/notebook ecosystem. A user-friendly interface, easy access to data from KCDC, and modern analysis software are of special interest. This contribution will discuss the service architecture, followed by a brief usage example.

Keywords: Astroparticle Physics · Open Data · Data Analysis · Jupyter

1 Motivation

The KASCADE Cosmic-ray Data Center (KCDC) [2] is a public data shop developed at the Institute for Nuclear Physics at the Karlsruhe Institute for Technology. KCDC provides access to reconstructed cosmic-ray data taken by the KASCADE [1] experiment.

We identified improvements and extension that will benefit our users:

- The data-sets can be very large, depending on the cut and selection criteria, and personal disk-space and bandwidth to download the desired data-set quickly becomes a bottleneck.
- Analyses usually rely on specialised frameworks and toolkits, some of which are very cumbersome to find, install, use and/or maintain.

One obvious solution to this is, instead of bringing data to the researcher, we bring the researcher to the data.

2 Design

2.1 Requirements

In order to find the best possible technology, we formulated the following requirements.

– Accessibility: we want people to actually use it, not yet another prestige project.
– Usability: the environment should provide all the of the tools necessary to conduct an analysis of KASCADE data.
– Administration: the entire software stack should be easy to install, maintain and update.
– Don't reinvent the wheel, most of the work has already been done.

2.2 Solution

Accessibility We decided to use the Jupyterhub / Jupyter Notebook [3] ecosystem for our analysis framework, as it already is very common analysis tool throughout all industries and research branches. Almost anyone how has done some data analysis in the last couple of year has come across or even used jupyter notebooks.

Useability Python is the current go-to language for data analysis and there's a plethora of third-party analysis packages. We also have to provide ROOT [5], as many example analyses still rely on it and it supplies the standard file format in particle physics.

Administration Notebooks allow arbitrary code execution by design, so it is sensible to isolate the notebook server of each user from one another and from the operating system. It shouldn't come as a surprise, that docker [4] was the best choice for containerization, scalability, resource management, ease-of-use, etc.

Though there is some custom code needed to attach Jupyter to KCDC, the majority of components are off-the-shelf industry standards. It is unlikely that we have to heavily invest into in-house development to keep these running or even extend and improve over time.

2.3 Architecture

Jupyter notebooks are a special kind of file format for cell-based, interactive programming. The standard user interface is implemented as a web service (notebook server); the important features in context of this document are the file-system-like contents manager and the notebook text-editor/execution environment ("kernel" in ipython jargon).

The notebook servers are docker containers created on demand with exclusive access by a single user each. They are proxied by another container running the jupyterhub software; the jupyterhub handles authentication and creation/deletion of notebook servers.

Choosing docker as virtualisation technology has many benefits, first and foremost process isolation, which is a good idea when your service requires arbitray code execution. Realizing the service as a docker stack, we not only gain

easy scalability for many users across multiple physical host machines, but also a text-based infrastructure description, which makes deployment and version control trivial.

For simplicity, users can use their KCDC credentials for authentication.

The contents manager has been extended for transparent access to the KCDC download area, and it is comparatively simple to add even more data shops, depending of course on the interfaces exposed by a data shop.

The current default notebook is configured with kernels for both Python and C++ (via the ROOT interpreter).

A graphical representation of the proposed analysis framework with KCDC integration can be found at Fig. 1.

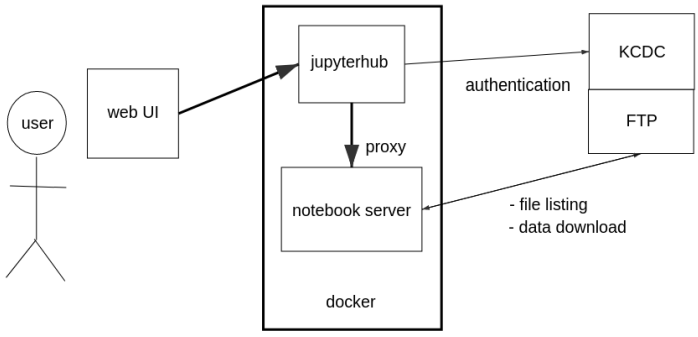


Fig. 1. A graphical representation of the proposed analysis framework with KCDC integration. Each user has their own containerized (via docker) analysis environment. The KCDC download area, where the users requested data-sets reside, can be accessed directly from the notebook interface to view or download to the notebooks dedicated work directory.

3 Example

This section is generated from the ipython notebook presentation at DLC 2020 [6], where it was used to showcase the capabilities of the analysis framework, and it is used now to demonstrate how easy it is to convert your analysis or presentation to a paper.

We also showed the importing of data from KCDC to the analysis framework in the original presentation, and although it is a key design aspect of the described framework, the ease-of-use of this feature doesn't lend itself to an enlightning description on paper, short of an uninspired screenshot or the mention of "There's a button for that".

Successfull acquisition of data is therefore presupposed in the following analysis example.

We are going to apply the KCDC example analysis to one of the preselected data-sets, for reasons of recreatability. After importing both the data file and analysis script, create a new notebook in the same directory.

[1]:
```
import os
from zipfile import ZipFile
ZipFile('KASCADE_SmallDataSample_wA_runs_0877-7417_ROOT.zip').
 ↪extractall()
os.listdir()
```

[1]:
```
['.ipynb_checkpoints',
 'KCDC_analyze_example.C',
 'slides.ipynb',
 'KASCADE_SmallDataSample_wA_runs_0877-7417_ROOT.zip',
 'info.txt',
 'events.root',
 'EULA.pdf']
```

Along with preselected data-sets, KCDC also provides a basic example analysis in C++ + ROOT. In order to run the KCDC example analysis, switch the notebook kernel to C++ and enter:

[1]:
```
.L KCDC_analyze_example.C
```

[2]:
```
run()
```

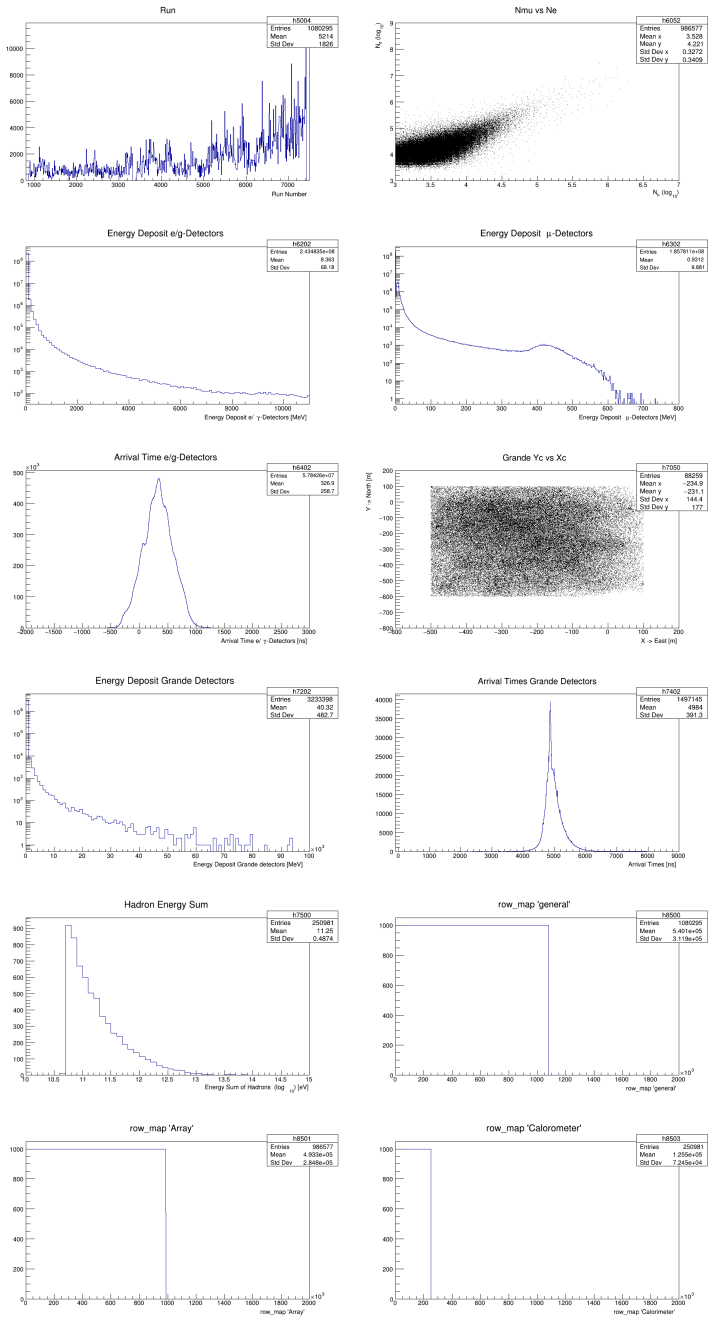
[2]:
```
Input file:events.root
KCDC-Entries read from files: 1080295
KCDCM-Entries:     1080295
Array Entries:     986577
Calor Entries:     250981
Grande Entries:    88259
General Entries:   1080295
KCDCN-Entries to be evaluated: 1080295
 processing event No: 0   of 1080295
 processing event No: 100000   of 1080295
 processing event No: 200000   of 1080295
 processing event No: 300000   of 1080295
 processing event No: 400000   of 1080295
 processing event No: 500000   of 1080295
 processing event No: 600000   of 1080295
 processing event No: 700000   of 1080295
 processing event No: 800000   of 1080295
 processing event No: 900000   of 1080295
 processing event No: 1000000   of 1080295
Entries survived:: 1080295 out of 1080295
general_id >0    :: 1080295
```

```
array_id >0     :: 986577
calorimter_id >0:: 250981
grande_id >0    :: 88259
(int) 0
```

Switch back to Python, and use PyROOT to open and display the result.

[1]:
```
import ROOT
f = ROOT.TFile('KCDC_Test.root')
keys = [_.GetName() for _ in f.GetListOfKeys()]
c = ROOT.TCanvas("foo", "bar", 1920, 1080*len(keys)//4)
c.Divide(2,len(keys)//2)
c.SetLogy()
pad = 0
logspectra = ['h6202', 'h6302', 'h7202']
for key in keys:
    pad+=1
    c.cd(pad)
    if key in logspectra:
        ROOT.gPad.SetLogy()
    f.Get(key).Draw()
c.Draw()
```

[1]:
```
Welcome to JupyROOT 6.20/04
```



The result of this example analysis shows, some statistics of the data set (ex. events over run number), some spectra (ex. energy deposit μ-Detectors), among other interesting things.

4 Outlook

We consider the integration of our data shop and analysis framework as a successfull technology demonstration. Going public and making it available for all KCDC users is the current goal, and we also plan to extend the functionality; it is comparatively simple to add more off-site data shops, there's the possibility to have multi-core and GPU support, and we need to evaluate more third-party analysis packages and notebook extensions. We also anticipate that user feedback is going to be another driving force behind long-term improvement of this service.

References

1. T. Antoni et al; The Cosmic-Ray Experiment KASCADE; Nucl.Instr. and Meth A513 (2003) 490-510
2. A. Haungs et al; The KASCADE Cosmic-ray Data Centre KCDC: Granting Open Access to Astroparticle Physics Research Data; Eur. Phys. J. C (2018) 78:741; https://doi.org/10.1140/epjc/s10052-018-6221-2
3. Thomas Kluyver and Benjamin Ragan-Kelley and Fernando Pérez and Brian E. Granger and Matthias Bussonnier and Jonathan Frederic and Kyle Kelley and Jessica B. Hamrick and Jason Grout and Sylvain Corlay and Paul Ivanov and Damián Avila and Safia Abdalla and Carol Willing and et al.; Jupyter Notebooks - a publishing format for reproducible computational workflows; Kluyver2016JupyterN 2016;
4. Dirk Merkel. 2014. Docker: lightweight Linux containers for consistent development and deployment. Linux J. 2014, 239; Article 2 (March 2014)
5. Rene Brun and Fons Rademakers; ROOT - An Object Oriented Data Analysis Framework; Proceedings AIHENP'96 Workshop, Lausanne, Sep. 1996, Nucl. Inst. & Meth. in Phys. Res. A 389 (1997) 81-86. See also http://root.cern.ch/
6. IV International Workshop "Data life cycle in physics", DLC-2020; https://indico.scc.kit.edu/event/806/