Jupyter-based service for JUNO analysis

Tao Lin[1,*] (on behalf of the JUNO collaboration)

[1]Institute of High Energy Physics, Chinese Academy of Sciences, Beijing 100049, China

Abstract. The JUNO (Jiangmen Underground Neutrino Observatory) is designed to determine the neutrino mass hierarchy and precisely measure oscillation parameters. The estimated data volume of raw data is about 2 PB/year. The event rate of reactor anti-neutrinos is about 60/day, while the event rate of background is about $O(10)$ Hz. The challenge is the event correlation during the analysis, where the background events could not be discarded. In order to use big data techniques to search for rare events, a Jupyter-based interactive service is developed for JUNO analysis.
In this paper, an overview of this service is presented. The infrastructure is based on Jupyter and Kubernetes, which provides the user interface and resource management. In order to integrate the data processing framework and big data techniques, an index file is used as an intermediate file, which points to the interested events. Data processing framework SNiPER is used to select the candidate of neutrino signals and produce the index file. Apache Spark is then used to process such index file repeatedly with data cached in memory. With the index file produced from Spark and the complete event data files, SNiPER is used to process them and produce the final physics result. At the end of paper, the test-bed is presented and the testing result is shown.

1 Introduction

The JUNO [1, 2] experiment, located in southern China, aims to determine the neutrino mass hierarchy and observe neutrinos from terrestrial and extra-terrestrial sources, including the supernova burst neutrinos, diffuse supernova neutrinos, geo-neutrinos, atmospheric neutrinos and solar neutrinos. It is about 53 km away from the Yangjiang and Taishan nuclear power plants.

Figure 1 shows the schematic view of the JUNO detector, where the innermost part is the central detector which is surrounded by a water Cerenkov detector. Not only the radioactivity backgrounds, but also neutrons introduced by cosmic-ray muons in the rock are heavily suppressed by layer of water. Instrumented with photomultiplier tubes (PMTs), the water Cherenkov detector can detect cosmic-ray muons. There is a 3-layer plastic scintillator top tracker above the Cherenkov detector, which provides precise and independent (cosmic-ray) muon track information.

The JUNO central detector is a 20 kt spherical volume of liquid scintillator (LS) with 35 m diameter instrumented with 18,000 20-inch PMTs and 25,000 3-inch PMTs. Neutrinos are captured by protons of the target via the inverse beta decay reaction which produces a

[*]e-mail: lintao@ihep.ac.cn

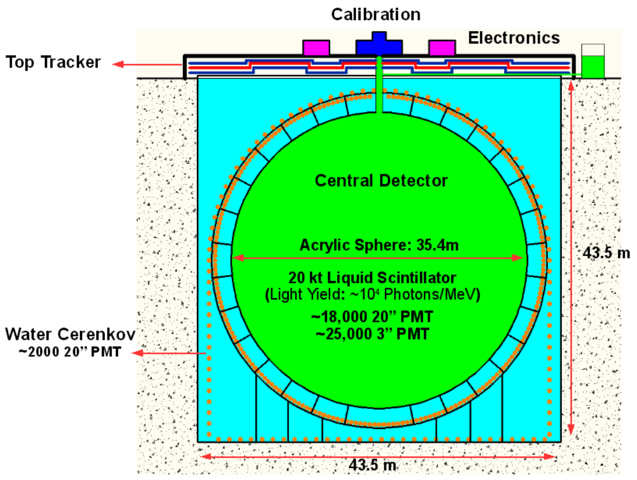

Figure 1. Schematic view of the JUNO detector

positron and a neutron. The time correlation between the prompt signal from the annihilation and ionization of the positron and the delayed signal from the gamma rays produced by the neutron capture is the crucial feature that makes it possible to detect neutrinos. Analysis of time correlations to separate neutrino candidates from background sources of coincident signals is crucial to the JUNO.

2 Motivation

The estimated data volume of raw data is about 2 PB/year. Compared to the rare neutrino signals, which is about 60/day, the rate of background is about $O(10)$ Hz. Accessing dataset many times may cause a drop of I/O performance due to the background levels. Meanwhile, the time correlation is still important. No events could be discarded.

In order to improve the efficiency to search for rare events in the huge background level, the big data techniques such as in-memory analysis are under investigation. Apach Spark [3] is chosen for its simplicity and easy-to-use nature. The physics quantities are loaded into memory and can be processed many times. After getting a subset of events, the SNiPER [4] framework is used to analyze the events with time correlation. Figure 2 shows the integration of ROOT [5] and interactive analysis. The underlying computing resources are shared between them. User could use both the SSH login with X server and the web browser to access the resources. In the web based analysis service, a middle-ware will be developed.

3 Designs

3.1 Infrastructures

The Jupyter-based service adopts Jupyter Notebook [6] and its next generation JupyterLab as the web-based user interfaces. The JupyterHub is adopted to provide groups of users to access their own dedicated computational environments and resources. The computational resources are then managed by Kubernetes [7]. The requests of resources from Jupyter will be done by the Kubernetes.

The key features of the service are:

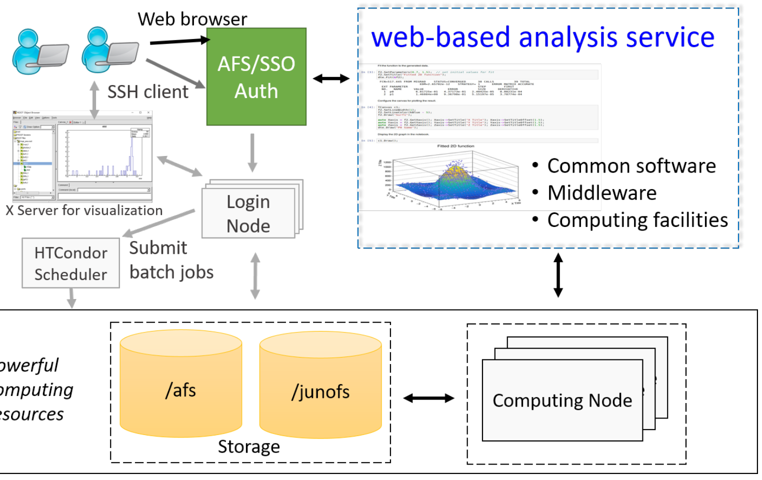

Figure 2. Schematic view of integration of ROOT and interactive analysis

- User-friendly. The users do not need to install the software themselves. Both the Jupyter software and JUNO software are deployed into the shared file system.
- Dedicated resources. The users could use the resources allocated for them.
- Scalablity. Benefit from Kubernetes, the users could request more resources.

In order to enhance the user experience, JupyterHub is customized including the Docker [8] images for users, limits of memory and CPU, and the mounted file systems. The Docker image is based on the distribution from IHEP. CernVM File System (CVMFS) [9] is configured by default in the images, so users could access the JUNO software on-demand.

3.2 Unified Event Data Store

In order to support the Jupyter Notebook, the normal cluster and the Spark cluster, a unified event data store is under development, as shown in Figure 3. The access to an event could be done by the direct access to the event data, or through the index files redirecting to the event data. The advantange of using the index file rather than using the whole set of event data is studied in other experiments, such as the AOD and Mini-AOD in LHC experiments, which allows fast selection and reduces the data size [10, 11].

In the index files, there are two major fields: the file index and the entry of the event in the target file. The other fields could be extended and used for the event reduction with big data techniques. All these physical quantities are extracted from the event data files, hence the size of each record is reduced. The contents for these index files could be varied along with the different big data techniques.

3.3 Analysis procedures

As shown in Figure 4, the procedures are:

1. Using ROOT/SNiPER to produce the primary index files from the event data.
2. Using the Spark to analyze the index files interactively. All the index files are cached in the memory. The final results are then stored into the user index files.

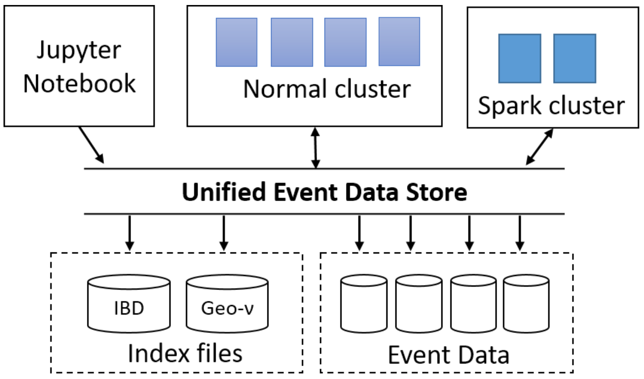

Figure 3. The unified event data store

3. Using ROOT/SNiPER to load the user index files and access the selected events directly. Finally, the plots and nutples can be shown in Jupyter.

In the first and third steps, SNiPER enables the time correlation analysis. In the second step, non-correlation analysis is done.

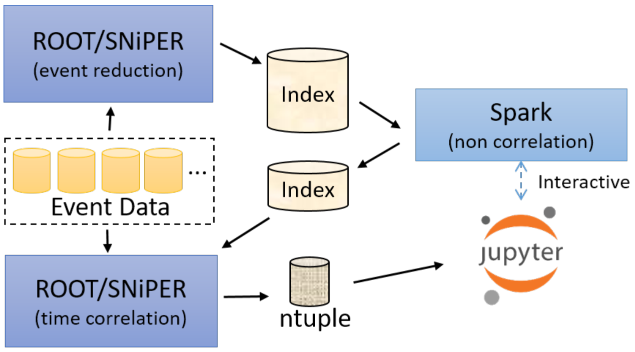

Figure 4. Analysis with Spark and SNiPER

4 Test-bed and Results

The test-bed is established in two rack-mounted servers with total 40 cores (Intel Xeon Silver 4114), which are managed by the Kubernetes. The JupyterHub is then started as a container inside the Kubernetes. In order to avoid the resource competition between Jupyter and Spark due to the limited resources, the Spark cluster is separate from the Kubernetes. The Spark cluster is built using 8 nodes with Intel Xeon E5-2630L v2 (12 cores, 24 threads).

Figure 5 shows an example to measure the data processing time in the test-bed. In this example, ROOT files are generated with different number of events and then they are processed with a ROOT script . The data files are processed using Spark with spark-root [12], with or without cache. The cache option will let spark cache all data in the memory. As shown in the table 1, there is about 10x speedup using spark with cache compared with ROOT.

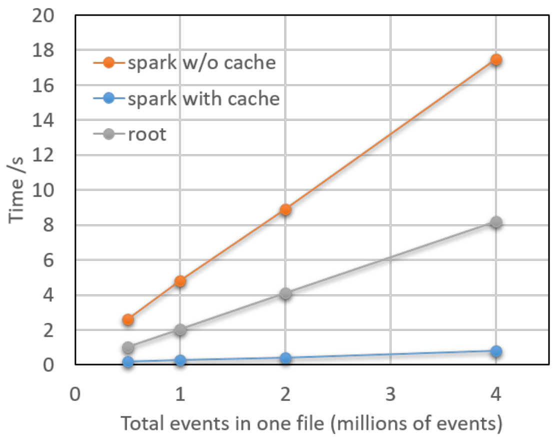

Figure 5. The example of measured performance in the test-bed

Table 1. Measured time of the example

Total Events/10^6	Time/s (root)	Time/s (spark w/o cache)	Time/s (spark with cache)
0.5	1.03	2.60	0.18
1	2.05	4.80	0.30
2	4.11	8.92	0.41
4	8.20	17.50	0.81

5 Conclusions

In this paper, a Jupyter-based service for the JUNO analysis is shown. By using the open-source software, such as Jupyter and Kubernetes, users could analyze data interactively on a dedicated computing node. In order to handle time-correlation analysis using big data techniques, an index file is used as intermediate file. The time correlation analysis is still done by the data processing framework SNiPER, while the event selection of single event is offload to big data techniques. A test-bed is also established at IHEP and used for studies of big data techniques for the JUNO.

The next work focuses on the application of big data techniques. One possible research is using the HDF5 file format for index files and then analyzing them using Spark. Another interesting research is using the PyRDF, which enables the Spark back-end automatically. As the C++ is still used in the SNiPER framework, a MPI based solution could be used to speed up the data processing.

Acknowledgements

This work is supported by National Natural Science Foundation of China (11805223), Joint Large-Scale Scientific Facility Funds of the NSFC and CAS (U1532258), the Strategic Priority Research Program of the Chinese Academy of Sciences, Grant No. XDA10010900.

References

[1] F. An et al. (JUNO), J. Phys. **G43**, 030401 (2016), `arXiv:1507.05613`

[2] Z. Djurcic et al. (JUNO) (2015), `arXiv:1508.07166`

[3] M. Zaharia, M. Chowdhury, M.J. Franklin, S. Shenker, I. Stoica, *Spark: Cluster Computing with Working Sets*, in *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing* (USENIX Association, USA, 2010), HotCloud'10, p. 10

[4] J.H. Zou, X.T. Huang, W.D. Li, T. Lin, T. Li, K. Zhang, Z.Y. Deng, G.F. Cao, J. Phys. Conf. Ser. **664**, 072053 (2015)

[5] R. Brun, F. Rademakers, Nucl. Instrum. Meth. **A389**, 81 (1997)

[6] *Project Jupyter*, https://jupyter.org/

[7] *Kubernetes (K8s)*, https://kubernetes.io/

[8] *Docker*, https://www.docker.com

[9] C. Condurache, I. Collier, J. Phys. Conf. Ser. **513**, 032020 (2014)

[10] W. Ehrenfeld et al. (ATLAS), J. Phys. Conf. Ser. **331**, 032007 (2011)

[11] G. Petrucciani, A. Rizzi, C. Vuosalo (CMS), J. Phys. Conf. Ser. **664**, 7 (2015), `1702.04685`

[12] V. Khristenko, J. Pivarski, *diana-hep/spark-root: Release 0.1.14* (2017), `https://doi.org/10.5281/zenodo.1034230`