ISBN: 978-3-95450-232-5 ISSN: 2673-7000 doi:10.18429/JACoW-NAPAC2022-TUYE3

AN OPEN-SOURCE BASED DATA MANAGEMENT AND PROCESSING FRAMEWORK ON A CENTRAL SERVER FOR SCIENTIFIC EXPERIMENTAL DATA*

A. Liu†, W. Si, J. Callahan, S. Poddar, Euclid Techlabs, LLC, Bolingbrook, IL, USA
J. Gao, AJS Smartech, LLC, Naperville, IL, USA

Abstract

The ever-expanding size of accelerator operation and experimental data including those generated by electron microscopes and beamline facilities renders most proprietary software inefficient at managing data. The Findability, Accessibility, Interoperability, and Reuse (FAIR) principles of digital assets require a convenient platform for users to share and manage data on. An open-source data framework for storing raw data and metadata, hosting databases, and providing a platform for data processing and visualization is highly desirable. In this paper, we present an open-source, infrastructure-independent data management software framework, named by EuclidLIMS, to archive, register, record, visualize and process experimental data. The software was targeted initially for electron microscopes, but can be widely applied to all scientific experimental data.

INTRODUCTION

Particle beam accelerators and beamlines are used in a wide range of multidisciplinary research, including nuclear physics, advanced material physics, high energy physics, biophysics, and beyond. These facilities and instruments generate large amounts of valuable scientific experimental data, both for hardware diagnostics purposes and for scientific discoveries. The types of data generated in these experiments include images, text files, spreadsheets, proprietary file formats, and so forth. A lot of these data contain rich metadata information, such as the instrumentation name, model and settings, experimental conditions, purposes and configurations, and more.

With the rapid development of detectors and pre-analysis electronic systems, the volume and the speed of the data generation are both growing significantly. Keeping up with the explosive increase in scientific data creation has been practically impossible for many existing data management frameworks that are purely based on a local data and computing architecture, particularly for comparably smaller facilities where the infrastructure is still not fully developed yet. However, without a data management software, the scientific data collected can be easily unintentionally lost no matter whether it is stored on a portable storage disk, a personal or organizational computer, or on a local data server. Finding the data-of-interest from a sea of folders and files by filtering with the filename is rather cumbersome. Moreover, physical properties like aforementioned are prone to being lost, either by being left behind or damages.

It is well accepted that a data management software can mitigate the above problems with the proper usage of a well-managed data server and a powerful software. Furthermore, in order to deploy the software on computers with old operating systems (OS), which are prohibited to be connected to a public network (WWW) and only support local network connectivity. Therefore, the software not only needs to provide all the necessary functionalities for the data management, but also is required to be "miniature" such that it can be adopted by most of the experimental computers.

Another desired capability of the software is providing an online data analysis platform for the raw data collected. With the fast advancement of cost-effective computing hardware, many experimental facilities are now equipped with computing servers that are able to handle not only the data storage, but also resource-demanding computing jobs. Therefore, it is beneficial to be able to launch computationally heavy jobs on the server so the managed data can be analyzed in the background.

EUCLID-NEXUSLIMS

Euclid has been collaborating with a group at NIST to develop a infrastructure, project and facility-independent software framework for data management and processing. The Euclid software framework is based on the NexusLIMS [1] open-source program developed by the NIST collaborators, and was initially named "Euclid-NexusLIMS". LIMS stands for "laboratory information management system". During the development of Euclid-NexusLIMS, we fully respected the open-source software and FAIR principles. One of our main goals was to accommodate a broader spectrum of scientific data management requirements by taking advantage of an already demonstrated framework instead of reinventing the wheel.

The framework is made of several main components. The ones that are present in both the original NexusLIMS and our Euclid-NexusLIMS are a data *logger* graphical user interface (GUI) that can be installed on control computers that are used to control the instrumentation to take data, a database record and metadata generator *backend*, and a *frontend* with Web interface for browser-based user interaction. A computing module in Euclid-NexusLIMS allows users to run data analysis jobs with workload managers like Slurm or Snakemake [2]. The infrastructure-specific portions have

* Work supported by the U.S. DOE Office of Science under contract number DE-SC0021512.
† a.liu@euclidtechlabs.com, ao@aoliu.tech

ISBN: 978-3-95450-232-5 ISSN: 2673-7000 doi:10.18429/JACoW-NAPAC2022-TUYE3

been removed to be conveniently deployable at virtually all facilities. We then added support for more data file formats, such as TIFF and PCD files, such that more scientific data can be supported in the preview generation.

The original NexusLIMS was designed such that the whole framework does not depend on a public network (WWW). The required network connections are: a local area network (LAN) connection between the *logger* and the *backend*, a LAN connection between the servers hosting the *backend* and *frontend*, and a LAN connection between the user computer and the server hosting the *frontend* service. The *backend* and the *frontend* can also be hosted on the same server, which does not require the LAN connection in between. This design was made mainly due to the concern that the NIST safety policy does not allow these servers to be connected to the WWW.

The *backend* and *frontend* could both be hosted on a cloud computing platform, such as the Google Cloud Platform (GCP) and the Amazon Web Services (AWS), embracing online cloud services. With the new framework, users can directly save and back up data files to the cloud storage, preprocess and postprocess data online, and view and download data from anywhere that has Internet access.

To distinguish the two variations of Euclid-NexusLIMS, the cloud-compatible version is named "EuclidLIMS", versus the local-only version "Euclid-NexusLIMS-local". In Fig. 1, the workflow of Euclid-NexusLIMS-local and EuclidLIMS are presented.

EUCLIDLIMS

EuclidLIMS, the cloud version, a user reserves an experimental "session" for a certain machine in the facility on an online calendaring system like a Google Calendar prior to the experiment. The user then starts the session via using the *logger* miniature GUI that is written with Python 3.4 and modules compatible with the Python version. This allows the *logger* to be installed and run on old OS such as Windows XP, which was the oldest Windows OS EuclidLIMS was tested with. Note that it is not necessary to use *logger* on the computer that is controlling the data taking instruments, as long as the computer that runs the *logger* can read and write in the network file system (NFS) that the data files are saved to. The network address of the *logger* computer is used to identify itself in a predefined computer database, such that the correct session information can be found in the online calendering system.

The *logger* connects to the cloud's MySQL database (as used in our demonstration with the GCP) with Flask and writes an entry to it with the session information, including the datetime for the start and end of the session, experimenter name, session notes, *logger* computer name, and so forth. It also supports richer metadata with a more sophisticated calendaring system, where information like department name, project code, etc. can be recorded for the session.

The *logger* also handles the raw data file uploading from the NFS to the cloud storage buckets. This requires the user to be added to the GCP project that the buckets are associated to and the user authenticates him/herself through a Google Authentication browser page.

The *backend* deployed on the GCP watches for newly added database entries and the associated data files uploaded through the storage buckets. It is responsible for extracting metadata information from the raw data files, such as the camera configuration from a *.zvd* file generated by a Zivid Two camera, and generating thumbnail(preview) images for the *frontend*. This is enabled by deploying functions in *Google Functions*, which is a function-as-a-service (FaaS) provided by the GCP and allows for very high scalability. The function watches over a cloud bucket and is triggered by the creation of new files from users, and then runs and saves the output files in a separate storage bucket.

The *backend* also generates a "record" for the *frontend* for each experimental session. The record contains information that will be rendered for users who need to query the record database with keywords like the datetime, name, etc. Storage bucket information is also present in the record such that the *frontend* can find the locations of the data files for user downloads. The tasks of record generation and posting to the *frontend* via HTTP are enabled with the Google *App Engine*, in which a *cron* job-like periodic task execution is allowed. The *backend* regularly checks the records and identify the new ones that need to be processed and handles the tasks itself without intervention.

The *frontend* is a Web-based data visualization and operation tool designed to work with the records generated by the *backend*. The original codes were developed by NIST [3], and then adopted by our NIST collaborators for NexusLIMS, which was referred to as NexusLIMS-CDCS. It is containerized and can be deployed using *docker* and *docker-compose* either on the same server where the *backend* is running, or a dedicated separate Web server. It is set to always restart with the system when after it crashes. It is developed on top of the *Django* framework and provides extensive *REST* APIs. It renders the records uploaded by the *backend* in XML formats based on a pre-defined *schema*. We made some modifications to NexusLIMS-CDCS to deploy it in a virtual machine (VM) provided by the *compute engine* of the GCP, and refer to it as EuclidLIMS-CDCS for clarity. Two major views displayed to the user by EuclidLIMS-CDCS are shown in Fig. 2, where (a) is the overview of all records matching the query criteria and (b) shows the individual record for a single experimental session.

The *computing* module in EuclidLIMS, with a goal of supporting both *exploratory and interactive* and *intensive and batched* data processing. Both of them are done on the cloud. For exploratory data analysis, users are allowed to launch interactive sessions in *JupyterHub* to do "live scripting" with the data already in the storage buckets. The JupyterHub is deployed on the GCP using the Google Kubernetes Engine (GKE), which allows automatic scaling. Each GKE *pod* (a user computing environment) is spawned from the same docker image. Users can log in with Google credentials and perform notebook-based data analysis with Python, R, C++,

ISBN: 978-3-95450-232-5 ISSN: 2673-7000 doi:10.18429/JACoW-NAPAC2022-TUYE3

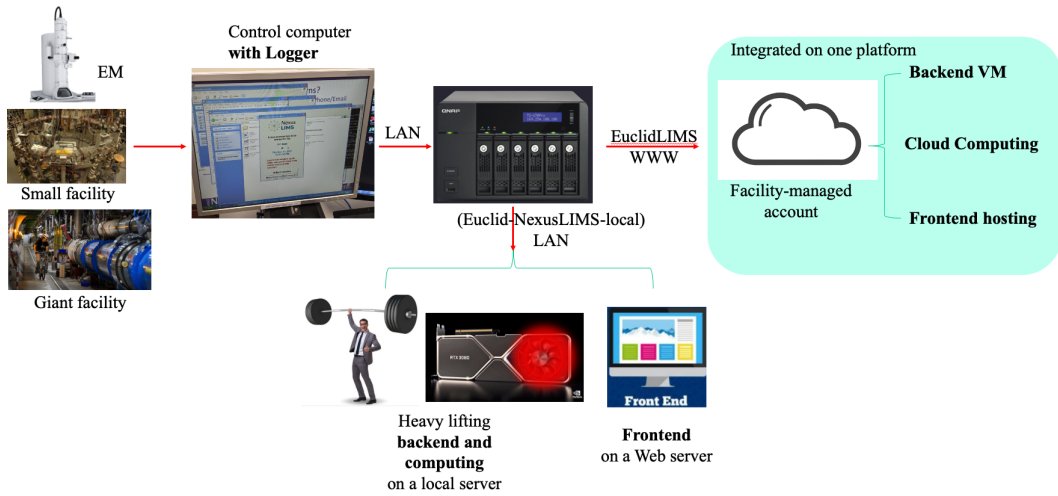

Figure 1: Scientific data management workflow with our two software variations: Euclid-NexusLIMS (local version) and EuclidLIMS (cloud version).

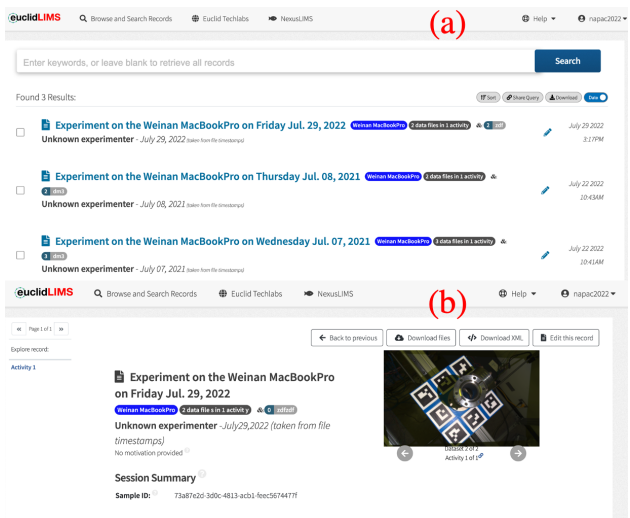

Figure 2: Two major views displayed to the user by EuclidLIMS-CDCS. (a): overview of all records matching the query criteria; (b): an individual record for a single experimental session.

Julia, etc. Since the raw data is contained in the buckets on the GCP, the data transfer is straight-forward and fast within the same Google platform.

For batch computing jobs and workflow management, we temporarily adopted an open-source workflow management tool, *snakemake*, which follows the same philosophy as make which provides more flexibility using python-like syntax. With proper configuration of job descriptions and scripts, users can launch batch processing jobs with a single command in the terminal. Jobs are sent over to the auto-provisioned virtual machines on the cloud. Upon the completion of jobs, the provisioned virtual machines are automatically shut down to save cost. The cloud platform generally offers a wide range of configurations of computing resources, users can configure their jobs depending on the concrete tasks to minimize the runtime without actually paying the actual price of the hardware. This is a reflection of the in-demand billing model of the cloud platform and one of the biggest advantages of EuclidLIMS.

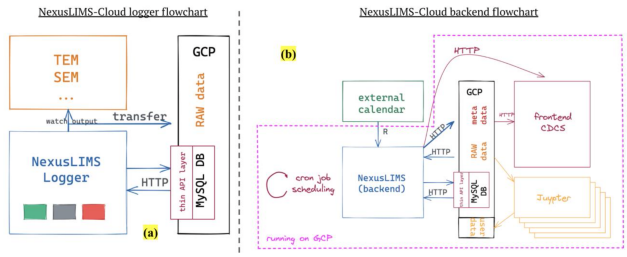

Figure 3: Block diagrams showing the EuclidLIMS framework.

The structure of the EuclidLIMS framework is shown in Fig. 3 with block diagrams.

CONCLUSION

A software framework EuclidLIMS for scientific data management and processing on cloud computing platforms has been created and demonstrated. The framework has components that are designed to allow for convenience in the framework deployment, user usage, and increased scalability.

ACKNOWLEDGEMENTS

The author would like to thank the whole project group at Euclid, especially Weinan Si for his invaluable contributions to the code development. The authors also thanks the NIST collaborators June Lau and Josh Taillon for sharing their knowledge and codes, and our Northwestern customers Roberto Reis and Laura Bartolo.

ISBN: 978-3-95450-232-5 ISSN: 2673-7000 doi:10.18429/JACoW-NAPAC2022-TUYE3

REFERENCES

[1] J. Taillon *et al.*, "NexusLIMS: Leveraging Shared Microscopy Resources for Data Analysis with a Configurable Laboratory Information Management System," *Microscopy and Microanalysis*, vol. 26, no. 52, 2020. doi:10.1017/S1431927620023314

[2] https://snakemake.readthedocs.io/en/stable/index.html

[3] https://www.nist.gov/itl/ssd/information-systems-group/configurable-data-curation-system-cdcs/about-cdcs