ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THMPA03

A SIMULATION SYSTEM FOR THE EUROPEAN SPALLATION SOURCE (ESS) DISTRIBUTED DATA STREAMING

C. Reis†, R. Borghes, G. Kourousias, R. Pugliese
Elettra Sincrotrone Trieste, Basovizza, Italy

Abstract

European Spallation Source (ESS), the next-generation neutron source facility, is expected to produce an immense amount of data. Various working group mostly associated with the European Union (EU) project Building a Research Infrastructure and Synergies for Highest Scientific Impact in ESS (BrightnESS) aim at developing solutions for its data-intensive challenges. The real-time data management and aggregation is among the top priorities. The Apache Kafka framework will be the base for ESS real-time distributed data streaming. One of the major challenges is the simulation of data streams from experimental data generation to data analysis and storage. This paper outlines a simulation approach based on the DonkiOrchestra data acquisition and experiment control framework, re-purposed as a data streaming simulation system compatible with ESS-KAKFA infrastructure.

INTRODUCTION

Elettra Sincrotrone Trieste is a multidisciplinary international research center specialized in generating high quality synchrotron and free-electron lasers (FEL) light and applying it in materials and life sciences.

The electron storage ring Elettra provides state-of-art techniques to lead experiments in physics, chemistry, biology, life sciences, environmental sciences, medicine and cultural heritage. It is the only third-generation synchrotron radiation source in the world that operates routinely at two different electron energies, i.e., 2.0 GeV for enhanced extended ultraviolet performance and spectroscopic applications, and 2.4 GeV for enhanced x-ray emission and diffraction applications. Currently 26 beamlines utilize the radiation generated by Elettra source.

The FEL FERMI light source has been developed to provide intense and fully coherent radiation pulses in the ultraviolet and soft x-ray range. The peak brightness of about 6 orders of magnitude higher than third generation sources generated by its single-pass linac-based FEL allows the performance of time-resolved experiments based on coherent diffraction imaging, elastic and inelastic scattering, photon and electron spectroscopy and transient grating spectroscopy. FERMI can operate in the 100-40 nm energy region in the initial phase and down to 10 nm in a subsequent phase. Currently 6 versatile experimental stations carry out outstanding research in diverse fields and disciplines.

The EU-founded BrightnESS project [1] aims at support the construction of the ESS in key technical areas and in-kind coordination. Elettra has been introduced and integrated to the main workload of BrightnESS Work Package 5 [2] which has the objective to maximize the scientific output of the ESS by enabling real time processing data taken on ESS instruments.

EXPERIMENTS IN NEUTRON AND ELECTRON FACILITIES

Each scientific setup is unique, not static and is periodically upgraded with new instrumentation, so a constant evolution of software and control systems technologies is necessary.

Computational Requirements

Neutron and electron facilities have demanding computational requirements; not only for their accelerators but also for their experimental stations and laboratories. Since these facilities are in a constant upgrade and change (as science always does) the computational systems require scalability and should allow for easy customization. Naturally such systems should permit concurrency as in parallel data processing, storage and data transfer. The latter requires intelligent and efficient architectures for transfers with minimal overheads. As expected, advanced workflow systems are used to facilitate communication, control and data flow.

Workflow Systems

A workflow system for the above-mentioned scenario should provide an infrastructure for the set-up, performance and monitoring of a defined sequence of tasks, arranged as a workflow application. From the workflow point of view, scientific experiments consist of a sequence of specifics tasks that can be organized in many different ways according to the experiment requirements. Schematically, an experiment sequence is composed by three consecutive phases: planning, collection and closeout. During the planing phase the sequence of the operations to be performed as well as its priority level should be set up. Once the planning phase is finished the system starts a sequence of triggers to communicate the operations to execute its tasks. Once a task is completed the current operation notifies the system, then proceed with the new task, and so on. Finally, in the closeout phase the system should perform any needed closing procedure to ensure all the operations were done.

† carlos.reis@elettra.eu

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THMPA03

Scripting and Control Systems

Thousands of devices and equipment must be remotely controlled and monitored from different locations like control room and beamline end-stations. Various local area networks connect networked devices, workstations distributed around the facility, and many servers. Distributed software frameworks based on special communications protocols, such as TANGO [3] or EPICS, realize advanced distributed control solutions.

DONKIORCHESTRA

DonkiOrchestra (DO) is a workflow management framework for end-station software development. It can provide logical organization by sending a train of software triggers where each trigger activates some action according to different priority levels [4]. This allows for concurrency and map-reduces strategies.

As in a symphonic orchestra the system is composed by a Director and multiple independent Players. Each Player belongs to a Priority group and has specific task to execute. The Director conducts the experimental sequence by sending a train of software triggers to the Players. For each step of the experimental sequence, a trigger signal is sent to the Players with the highest Priority 0, then to the group of Players with Priority 1 and so on. Each Player executes its task upon the arrival of the trigger and send back to the Director an acknowledge event. A simplified schema of the explained architecture is shown in Fig. 1.

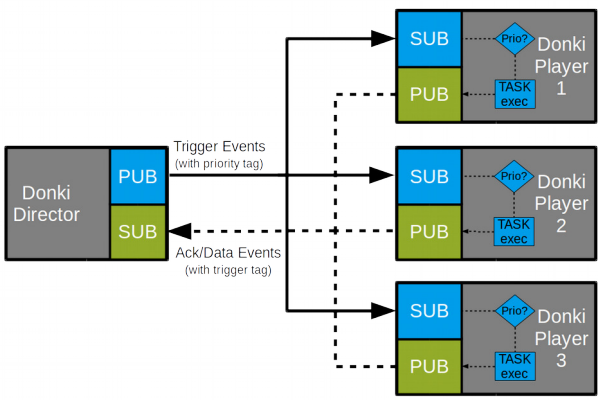

Figure 1: DonkiOrchestra schematic architecture.

DonkiOrchestra in Elettra and FERMI

As Elettra and FERMI beamlines consist of a complex distributed network of devices (e.g. sensors, detectors, motors, etc), a distributed control system approach has been used to develop DonkiOrchestra in its scope using TANGO [5]. The Director and Player of DO are independent software components (i.e., servers, scripts or any Python object) distributed on different computers connected through an Ethernet network. A goal is that it allows for reusability and simplicity for most of the system and its components. Due to its asynchronous I/O model, ZeroMQ was the chosen messaging system. It fits the need of having a scalable distributed application and also maximizes the opportunity of performing parallel tasks.

DonkiOrchestra in WP5

DonkiOrchestra has been re-purposed to be used with Kafka-ESS [6] technologies aiming at providing rapid scripting of complex scenarios of data stream generation and processing for simulation purposes. The framework became TANGO independent and a new information system based in TCP/IP protocols has been developed in order to establish the communication between the Director and the Players. ZeroMQ remains as the messaging system between the scheduler and Players. This system should allow for partial testing and development of the data aggregation software. A representation of the explained communication architecture is show in Fig. 2.

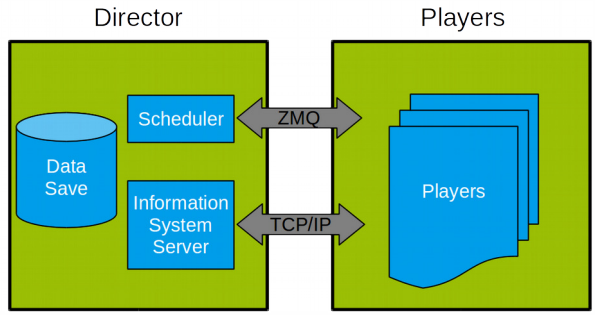

Figure 2: DonkiOrchestra communication schema.

Component Technologies

Tango Controls is an object-oriented, distributed control system framework which defines communication protocol API as well as provides a set of tools and libraries to build softwares control systems. Tango Controls is robust and easy to use toolkit and operating system independent that supports C++, Java and Python for all its components.

The Director and Player of DO are independent software components (i.e., servers, scripts or any Python object) distributed on different computers connected through an Ethernet network. A goal is that it allows for reusability and simplicity for most of the system and its components.

ZeroMQ is a distributed messaging system [7]. It has an asynchronous I/O model that fits the need of having scalable distributed application. It has a score of language APIs and runs on most operating systems. Different network topologies can be developed according to the ZeroMQ patterns. In the request-reply pattern ZeroMQ connects a set of clients to a set of services, defining a service bus topology. The publish-subscribe pattern connects a set of publishers to a set of subscribers, i.e., a data distribution tree. A parallelized pipeline can be defined through the push-pull pattern that connects nodes in a fan-out / fan-in model that can have multiple steps and loops. Due to its internal threading model and an automatic message batching technique, ZeroMQ can

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THMPA03

perform better than conventional TCP applications in terms of throughput.

DATA STREAMS

The progress in hardware technology has made it possible for organizations to store and record large streams of transactional data. Such data sets which continuously and rapidly grow over time are referred to as data streams.

This topic is a relatively recent one and many facilities of this kind do not rely on stream-based solutions even if the data they produce fit this model. The first research papers on this topic appeared slightly under a decade ago, and since then this field has grown rapidly. Data streams are a computational challenge to data mining problems because of the additional algorithmic constraints created by the large volume of data. In addition, the problem of temporal locality leads to a number of unique mining challenges in the data stream case [8].

Relevant characteristics

Data streams sources are characterized by continuously generating huge amounts of data from non stationary distributions. Some relevant characteristics of data streams processing include:

- The data elements in the stream arrive on-line.
- The system has no control over the order in which data elements arrive, either within a data stream or across data streams.
- Data streams are potentially unbound in size.
- Once an element from a data stream has been processed, it is discarded or archived. It cannot be retrieved easily unless it is explicit stored in memory, which is small relative to the size of the data streams.

Data Management at ESS

Most modern instruments at neutron facilities are equipped with a large number of detector segments and easily produce nearly 1 GB or more of data in a single measurement; analysis and storage of these large amounts of data are crucial [9].

An experiment or sequence of experiments result in a large number of documents, simulation results, measurement data, analysis results and scientific papers. All of this has to be stored in a structured way such that it is easily accessed both by humans and by software as appropriate. The data management system at ESS expects to handle a total data volume of approximately 2 petabytes per year.

Expected Data Flow for a Neutron Experiment

The following list provides a typical data flow for a neutron scattering experiment [10].

- **Experiment Control:** The team of users configure the components of the instrument and sample environment using an experiment control system that interfaces with the neutron instrument components through the ESS EPICS network.
- **Stream:** Data are taken in event mode whereby the individual detector counts are tagged with useful experimental metadata to create a dataset. The list of event and metadata are aggregated in software and broadcast over a network in a continuous stream of data that external softwares systems can utilise.
- **Reduce:** The raw data are transformed and corrected from the base unit of the instrument to a data type that is scientifically useful and valid. The objective is to take the large volumes of data and process them in as near real time as possible.
- **Visualise:** The representation to the beamline users of a scientifically meaningful display of the corrected data.
- **Analyse:** A scientific model is generated in order to scientifically interpret the experimental data.

Data Types and Frequencies at ESS

As described before the data acquisition at ESS is based on event mode. It means that a measurement is time stamped so that the time of the event can be associated with the value of a global clock. Neutron data rate differs from metadata rate, therefore whenever a metadata value changes, the new value is recorded together with the time stamp.

The following data types are expected for data coming from timing system, neutron data from detectors, metadata from control boxes and other sources.

- **Global time:** the full global time will be stored at the start of each accelerator pulse, which occurs at 14 Hz approximately.
- **Neutron event data:** consists of the neutron position and the time the event was recorded. Data from a neutron beam monitor may also be recorded.
- **Metadata:** experiment data on users, team members, local contact; sample under study; proton pulse data; beam monitor counts, neutron flux; moderator temperature; chopper settings, measured speed and phase; position of detector banks; instrument settings, alignment on optical systems; sample position including motor axle positions; pressure, humidity, temperature; sample environment; user-supplied instrument control commands.
- **Image data:** are typically a picture from a CCD camera which integrates the number of photons resulting from a neutron radiography. The data is a 2D array of intensities. This data can be read after a number of neutron pulses, after each pulse or maybe even several times during a neutron pulse.

It can be very useful but at the same time very challenging the simulation of streams with such complex data types generated at different frequencies. Thus a tool as DonkiOrchestra can help in the task of providing such simulations in an suitable way.

DATA STREAMS WITH DONKIORCHESTRA

In order to demonstrate the potential of DonkiOrchestra as a data stream simulator, and the simplicity of

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THMPA03

implementing it, we have developed and implemented a series of complex simulation examples. These examples include concurrency, i.e., parallel heterogeneous streams in terms of data types and frequencies. Table 1 presents the characteristics of the simulated data streams:

Table 1: Simulated Data Streams Characteristics

Data Stream	Frequency	Data Type
A	1 Hz	scalar
B	50 Hz	2D image
C	1 Hz – 1 kHz	1D spectrum
D	1 kHz	3D array
E	10 Hz	mixed

Implementation

A graphical user interface was developed in order to provide simplicity on managing the role of Director and Players, as well as simplicity on implementing the simulations.

In order to implement such data streams in DonkiOrchestra logic, different Python scripts are going to play the role of Players in which each one will generate a data stream case.

The procedure of adding a Player consist in setting a name for it, importing the Python script containing the action it will play and, also setting a priority level for it. The highest level of priority, i.e., the actions that will be done first, is zero. The level of priority setting allows for parallel actions when two or more Players are set with the same level of priority. The system also allow for disabling a Player by setting its level of priority to -1.

A sequence of software triggers is sent to the Players according to its priority level. The amount of triggers is set also through the graphical user interface. Thus, the system is ready to start the simulation.

An acknowledge signal sent by the Players arrives to the Director so than it manages the work flow until the system complete all tasks.

Figure 3 show a general flow for implementation of data streams simulation using DonkiOrchestra.

CONCLUSION

This paper presents a simulation approach based on the DonkiOrchestra data acquisition and experiment control framework, re-purposed as a data streaming simulation system compatible with ESS-Kafka infrastructure. The challenge of simulating data stream from experimental data generation is implemented through DonkiOrchestra logic in order to demonstrate the potential of the tool and the simplicity of implementing it. The design choices of a powerful messaging system like ZeroMQ that maximizes the opportunity of performing parallel operations, a dynamic and portable language like Python and a fully configurable structure that permits a high degree of customization defines the strength of this software product.

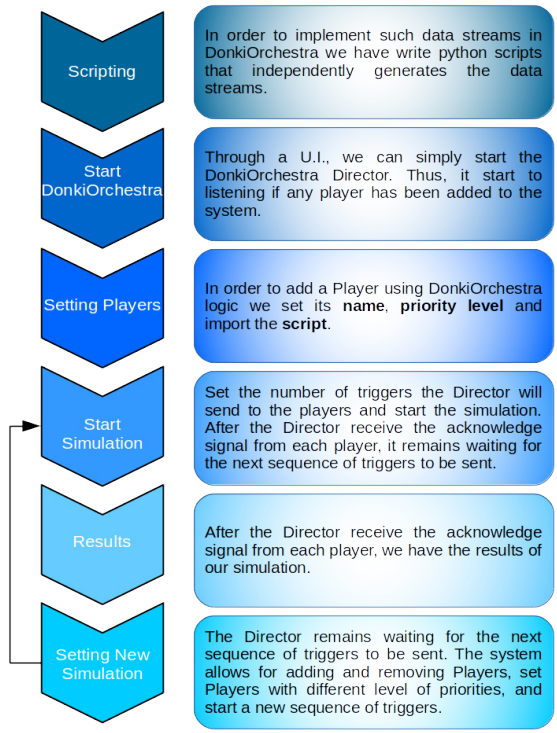

Figure 3: Implementation flow.

ACKNOWLEDGEMENT

The authors, members of the *Software for Experiments* team and the *Scientific Computing* team, thank the IT Group and the BrightnESS WP5 contributors. Among them special thanks for constructive discussions and feedback are due to Afonso Mukai and Tobias Richter. Moreover they acknowledge the importance of advanced technologies and open source software as well as international in-kind collaboration projects like that of BrightnESS.

REFERENCES

[1] BrightnESS, https://brightness.esss.se/about

[2] Work Package 5: Real-Time Management of ESS Data, https://brightness.esss.se/about/work-packages/work-package-5-real-time-management-ess-data

[3] Tango Controls, tango-controls.readthedocs.io/en/latest/contents.html

[4] R. Borghes and G. Kourousias, "DonkiOrchestra: a scalable system for data collection and experiment management based on ZeroMQ distributed messaging", in *Proc. 11th New Opportunities for Better User Group Software Conf. (NOBUGS'11)*, Copenhagen, Denmark, October 2016, paper 10.17199/NOBUGS2016.36, pp. 41-46.

[5] R. Borghes, G. Kourousias, V. Chenda, A. Gianoncelli, "Evolving a Labview End-Station Software to a Tango-Based Solution at the Twinmic Elettra Beamline", presented at 16th Int. *Conf. On Accelerator and Large Experimental Control Systems (ICALEPCS'17)*, Barcelona, Spain, October 2017, paper TUPHA208, this conference.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THMPA03

[6] A. Mukai *et al.*, "Development, testing and deployment of the ESS data aggregation and streaming software", poster presented in *11th New Opportunities for Better User Group Software Conf. (NOBUGS'11)*, Copenhagen, Denmark, October 2016.

[7] ZeroMQ, http://zeromq.org

[8] C. Aggarwal, Data Streams: Models and Algorithms. IBM T. J. Watson Research Center, NY, USA: Springer US, 2007.

[9] ESS, https://europeanspallationsource.se/data-management

[10] ESS, https://europeanspallationsource.se/sites/default/files/dmsc_workflow_ar2015_pdf_0.pdf