Open-Source Framework for Data Storage and Visualization of Real-Time Experiments

Preprint

Kumaraguru Prabakar, Nick Wunder, Nicholas Brunhart-Lupo, Courtney Pailing, Kristi Potter, Matthew Eash, and Kristin Munch

National Renewable Energy Laboratory

Presented at the 2020 IEEE Kansas Power and Energy Conference (KPEC 2020)
July 13–14, 2020

Conference Paper
NREL/CP-5D00-76264
July 2020

Open-Source Framework for Data Storage and Visualization of Real-Time Experiments

Preprint

Kumaraguru Prabakar, Nick Wunder, Nicholas Brunhart-Lupo, Courtney Pailing, Kristi Potter, Matthew Eash, and Kristin Munch

National Renewable Energy Laboratory

Suggested Citation
Prabakar, Kumaraguru, Nick Wunder, Nicholas Brunhart-Lupo, Courtney Pailing, Kristi Potter, Matthew Eash, and Kristin Munch. 2020. *Open-Source Framework for Data Storage and Visualization of Real-Time Experiments: Preprint*. Golden, CO: National Renewable Energy Laboratory. NREL/CP-5D00-76264. https://www.nrel.gov/docs/fy20osti/76264.pdf.

Conference Paper
NREL/CP-5D00-76264
July 2020

National Renewable Energy Laboratory
15013 Denver West Parkway
Golden, CO 80401
303-275-3000 • www.nrel.gov

Open-source framework for data storage and visualization of real-time experiments

Kumaraguru Prabakar, Nick Wunder, Nicholas Brunhart-Lupo, Courtney Pailing,
Kristi Potter, Matthew Eash, Kristin Munch
National Renewable Energy Laboratory
Golden, Colorado
Email: kumaraguru.prabakar@nrel.gov, nick.wunder@nrel.gov, nicholas.brunhart-lupo@nrel.gov, Courtney.Pailing@nrel.gov
kristi.potter@nrel.gov, matthew.eash@nrel.gov, kristin.munch@nrel.gov

Abstract—**Digital real time simulators (DRTS) are increasingly being used for the evaluation of power hardware and controller hardware in the laboratory prior to field deployment. Although DRTS are capable of simulating large models in real-time, it is challenging to visualize the results of large models without overdrawing or confounding the viewer. This paper provides an open-source framework for users to visualize their DRTS-based hardware-in-the-loop (HIL) experimental results in real-time. This proposed framework can be used by experimental test beds that can push data through an internet protocol based network. The proposed framework includes three main components. First, it includes the DRTS that generates and pushes the data to a relay. Second, it includes an application that serves multiple purposes, from data storage, testing, and translation of the data to a publisher/subscriber protocol. Finally, it includes libraries and applications that can be used to visualize the data by subscribing to the relay. This framework is available in open source, and it is tested using the HIL platform developed for testing advanced distribution management systems.**

***Keywords*—Advanced distribution management systems, controller-hardware-in-the-loop, digital real-time simulation, open-source framework, power-hardware-in-the-loop, real-time simulation, visualization, volt-VAR optimization.**

I. Introduction

Distribution systems are going through major technological advancements as a result of high penetrations of inverter-based assets, deployments of advanced metering infrastructure, and the integration of other smart grid technologies. The installation of these technologies in the distribution system without proper evaluation poses major risks for utilities. Vendors, utilities, and other research organizations are constantly creating new methods to evaluate these technologies safely, succinctly, and swiftly. These evaluation approaches help derisk the technology integration and help increase the technology-readiness level [1], [2].

In the literature [3], the evaluation methods (specifically for power applications) are typically classified into three major types: software-only modeling of the system and device under evaluation, hardware-only modeling of the system and device under evaluation, and evaluations based on hardware-in-the-loop (HIL). HIL can be further classified into two categories based on the device under evaluation: controller-hardware-in-the-loop (CHIL) and power-hardware-in-the-loop (PHIL). The research work presented here focuses on the HIL approaches, and the following information applies primarily to the HIL-based evaluation methods and might not necessarily apply to other evaluation methods.

In HIL-based evaluation, there are three main elements: the digital real-time simulator (DRTS), digital-to-analog conversion/analog-to-digital conversion, and the hardware (controller and power hardware) under evaluation. In the early years of research focused on enabling HIL, attention was given primarily to improvements in the time step capability and the number of nodes that can be simulated in a DRTS—and both have improved. At the time of writing this paper, commercially available DRTS using real-time phasor domain simulations can simulate systems with up to 30,000 single-phase nodes.

With these advancements in DRTS, users can now run system models with an exponentially increased number of nodes. Using DRTS creates the potential to simulate larger system models with reasonable time steps for longer periods without overruns. These improvements in DRTS-based simulations, however, create challenges in other aspects of CHIL- and PHIL-based experiments, such as in improving the accuracy [4] of HIL experiments, improving the stability [5] of HIL experiments, and creating function blocks [6] for the safe operation of experiments using high-power equipment ($\sim$1 MW).

These advancements also created other challenges: how to properly store these large amounts of data, visualize them, and provide detailed insights into the evaluation of the device under test and its impacts on the system under study. Visulaziation tools have been proposed in the [7], [8], but they were not built for experimental test beds and may not be able to handle large data sets that can be generated in an experimental test bed. The work presented here aims to solve these challenges by developing solutions using a proper software approach. Using the proposed approach, data from experiments can be stored

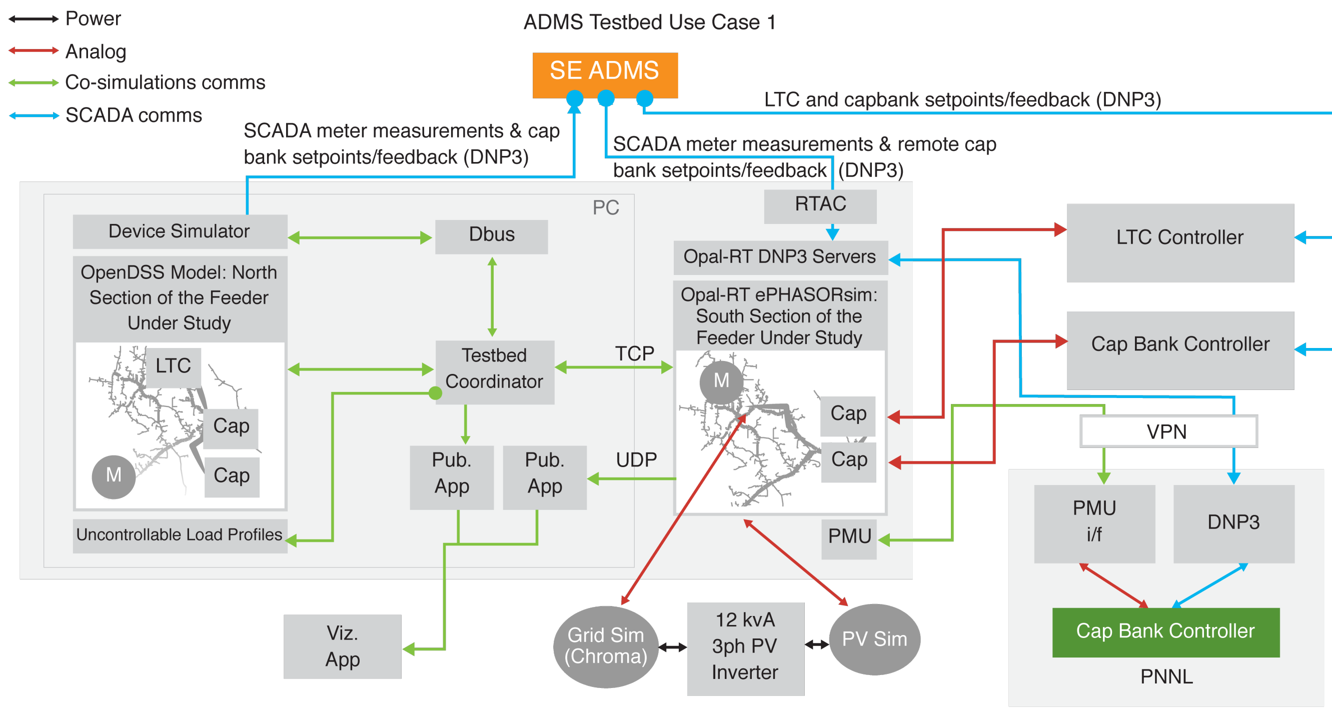
Power
Analog
Co-simulations comms
SCADA comms
ADMS Testbed Use Case 1
SE ADMS
LTC and capbank setpoints/feedback (DNP3)
SCADA meter measurements & cap bank setpoints/feedback (DNP3)
SCADA meter measurements & remote cap bank setpoints/feedback (DNP3)
RTAC
PC
Device Simulator
Dbus
Opal-RT DNP3 Servers
LTC Controller
OpenDSS Model: North Section of the Feeder Under Study
Opal-RT ePHASORsim: South Section of the Feeder Under Study
Cap Bank Controller
LTC
Testbed Coordinator
TCP
M
Cap
Cap
Cap
Cap
M
VPN
Pub. App
Pub. App
UDP
PMU i/f
DNP3
Uncontrollable Load Profiles
PMU
Viz. App
Grid Sim (Chroma)
12 kVA 3ph PV Inverter
PV Sim
Cap Bank Controller
PNNL

Fig. 1. Controller HIL and PHIL test bed setup used to evaluate the developed data storage and visualization tool

appropriately and visualized properly, and insights into models are made possible that were once reserved for non-real-time software simulation. The work presented here addresses these challenges associated with data storage and visualization using the recent development in open source publisher-subscriber model.

The solution provided here is agnostic to the DRTS used, controller under evaluation, or power hardware under evaluation. The solution was built using open-source tools, and it is made available in open source for researchers to use in their evaluation setup. The developed solution uses publisher-subscriber protocol to relay the data to multiple subscribers. This can be replaced with more secure protocols depending on the end-user requirements. This paper includes three major contributions: (1) the development of an open-source application to push data from raw data packets to a data relay (namely, the ØMQ [9] message bus); (2) the standardization of the data packets generated that need to be stored agnostic to the DRTS experiment platform; and (3) the development of an open-source application to visualize the data.

II. Experimental Setup

HIL experiments are becoming critical in evaluating the performance of a control hardware, such as a microgrid controller [3], advanced distribution management system (ADMS) [10], or power device controller such as an inverter controller [11] or motor drives [12]. PHIL experiments are run to test the performance of power hardware during power system dynamic events [13] or power system transient events [14]. Good real-time visualization of the experiments are crucial for proper debugging of the experiments and is becoming a crucial component in HIL evaluation. The following subsections explain the setup used for the HIL experiments and the data storage and visualization challenges faced during the experiments.

A. Hardware-in-the-Loop Setup

The experimental setup shown in Fig. 1 is a vendor-neutral evaluation platform. The goal of this test bed is to evaluate ADMS applications using the CHIL and PHIL setup. More information on the ADMS test bed setup is presented in [15]. Following are important components of the HIL setup that generate data that need to be stored and visualized.

1) Real-Time Simulation of the Distribution System Under Study: The first component in the HIL setup is the DRTS that simulates the power system under study. The setup here for the DRTS uses a hybrid cosimulation approach comprising an off-the-shelf dynamic simulator ePHASORSIM and the quasi-static time-series simulator OpenDSS. The multi-thousand-node distribution feeder model is split into two sections: the first section is simulated in OpenDSS, and the second section is simulated in ePHSORSIM. The two simulators exchange voltage and current information to run a hybrid cosimulation. More information on this cosimulation is available in [16], [17].

The crucial takeaway is the amount of data generated in real-time in both digital simulation tools. Both tools are superior in modeling a distribution system feeder, but they lack advanced real-time data storage and data visualization capabilities. In addition, the data need to be stored in real-time, which introduces additional complexity.

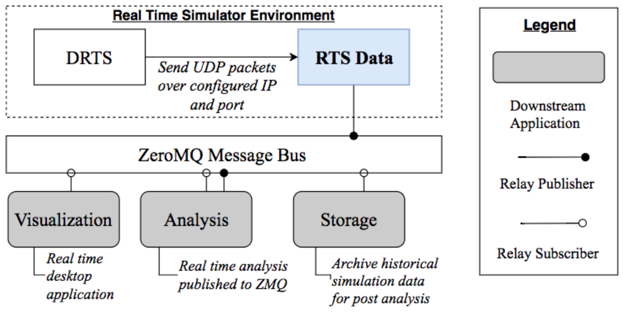
Real Time Simulator Environment
DRTS
Send UDP packets over configured IP and port
RTS Data
ZeroMQ Message Bus
Visualization
Analysis
Storage
Real time desktop application
Real time analysis published to ZMQ
Archive historical simulation data for post analysis
Legend
Downstream Application
Relay Publisher
Relay Subscriber

Fig. 2. Block diagram of the data path and the applications accessing the published data

2) Hardware Under Test: The setup used in this paper includes both controller hardware and power hardware under test. Three controllers are used in the CHIL setup: an off-the-shelf load tap changer (LTC) controller, an off-the-shelf capacitor bank controller, and the ADMS under study. The details of the ADMS are explained in the next subsection. For both controllers (the LTC controller and capacitor bank controller), the voltage at the bus was sent to the controller devices through analog channels. The LTC controller and the capacitor bank controller make set point decisions based on the terminal voltage. For example, the capacitor bank controller decides whether the capacitor banks should be turned on or off based on the terminal voltage. Similarly, the LTC adjusts the transformer position based on the terminal voltage it observes. One 12-kVA, three-phase photovoltaic (PV) inverter was used in the PHIL setup. This PV inverter is indicative of the inverters installed in the field.

3) Advanced Distribution Management System Under Test: In this test bed, a vendor-provided commercial ADMS was under test. Using this test bed to evaluate the performance of the ADMS enables derisking the field installation of the ADMS. As a first use case, the volt/volt ampere reactive optimization application of the ADMS was under evaluation[15]. The ADMS acts as the Distributed Network Protocol 3 (DNP3) client and connects to the DNP3 server devices modeled in the DRTS and the devices in HIL. This enables the ADMS to monitor the state of the distribution feeder and control the devices.

B. Challenges in Data Storage and Visualization

HIL experiments may need to be run for multiple hours and this results in the generation of a large data set, storing this data and visualizing this large data set pose a challenge. Most DRTS have internal data storage capabilities that are limited to a certain data size and are limited when the size of the model running in real-time is larger than a certain limit. Thus, the ideal data storage solution should have two important characteristics: (1) the data storage application should be stand-alone and separate from the real-time computation, and (2) visualization tool should be independent of the data storage application and should not interfere with the data storage process. The approach should be computationally simple. The approach presented here targets these necessary characteristics. The next section presents details on the approaches used to develop such an application.

III. Data Streaming Pipeline

DRTS are designed to perform computations in real-time, but they are not necessarily the best tools for data storage or real-time visualization of running experiments. For this reason, researchers spend countless hours waiting for an experiment to complete, then look at their data only to discover that there was an issue in the first second of the simulation. The researcher makes a minor change, starts the simulation again, waits for the simulation to complete, and discovers another error. Researchers are limited in both the volume of data and in the variety of visualizations available in existing DRTS software to debug experiments in real-time. This debugging cycle could be dramatically reduced if a tool existed to visualize experiments in real-time regardless of the time step of the simulation and the amount of data generated by the experiments.

To facilitate real-time visualization and analysis of these simulation data, a data-streaming pipeline was developed (Fig. 2). The DRTS supports the use of Ethernet cards to produce User Datagram Protocol (UDP) packets of simulation data. DRTS in general are capable of supporting multiple Ethernet cards pushing a variety of simulation data at variable frequencies. Each Ethernet card can be configured by researchers to push a specific set of simulation data at a fixed frequency for the duration of the simulation.

Once the UDP packets are sent by the real-time simulator, additional software is required to store and visualize these data. Data from the DRTS simulations flow through three main stages. First, data originate from a running DRTS simulation. These data are then picked up by the second stage in the pipeline: a desktop data-streaming application that relays DRTS data to any number of downstream applications. The third and final pipeline stage includes these downstream applications, which might include real-time visualization applications, real-time analysis applications, or data storage tools.

An open-source tool—designed to run on a Windows hosts and listen to data sent over DRTS Ethernet cards—was developed to facilitate capturing, processing, and streaming these simulation data in real-time. The Real-Time Streaming (RTS) data application is the second stage in the overall data pipeline described in this paper. The core functionality is to stream data as they are generated by the DRTS software to a message broker for other downstream applications, such as data-archiving tools or real-time visualizations. The RTS Data tool is configured to accept UDP data from the DRTS and relay these data, using Transmission Control Protocol, to a server hosting a message broker. Configuration is defined by DRTS Internet Protocol (IP) address, port, message broker IP address and port, frequency of UDP packets, as well as the size of and endianness of each datum. One instance of the application is designed to relay data from one DRTS Ethernet card to the message broker. Multiple instances of the application are required to run if the DRTS makes use of more than one Ethernet card; a single application instance will be considered in future work. Fig. 3 shows the block diagram of the completed setup and provides more details on the data flow from the experimental setup of the visualization application.

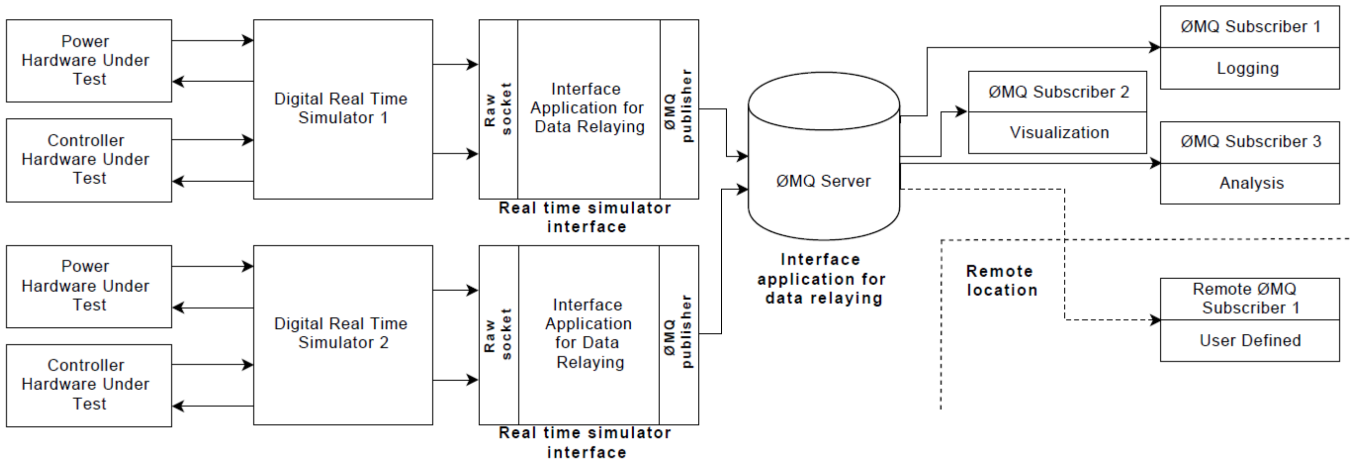
Power Hardware Under Test
Controller Hardware Under Test
Digital Real Time Simulator 1
Raw socket
Interface Application for Data Relaying
ØMQ publisher
Real time simulator interface
Power Hardware Under Test
Controller Hardware Under Test
Digital Real Time Simulator 2
Raw socket
Interface Application for Data Relaying
ØMQ publisher
Real time simulator interface
ØMQ Server
Interface application for data relaying
ØMQ Subscriber 1
Logging
ØMQ Subscriber 2
Visualization
ØMQ Subscriber 3
Analysis
Remote location
Remote ØMQ Subscriber 1
User Defined

Fig. 3. Block diagram of the completed setup

A. *Real-Time Streaming Data Software*

RTS Data is a desktop application enabling researchers to collect, stream, and store data produced by DRTS. The RTS Data tool is developed using modern Web technologies designed to look and feel like a modern single-page Web application.

RTS Data includes three main components: collect (Target/RTS), stream (Relay), and store (Archive). Each component is designed to be configurable as well as to run in test mode without the need to establish a connection to a running real-time-streaming simulation. These components and the setup are shown in Fig. 3. The code for the RTS Data is open source and is available for download at [18].

The initial component collects data from a configured running DRTS simulation, or target. The target is configured to send UDP packets with raw bytes over its network interface. This byte stream is either big endian or little endian encoded floating point numbers DRTS simulation data in 4 byte, or 32 bit chunks. RTS Data reads this raw byte stream, splits the stream into 4 byte chunks, converts each chunk to a floating point number, then pushes the converted floating point number to an array. After the raw byte stream is converted, the array of floating point numbers is published to an internal message broker for other modules of RTS Data to subscribe.

In order to provide downstream application with DRTS simulation data, the RTS Data application is configured to stream all converted data to a centralized message broker service. In this workflow, ZeroMQ [9] was adopted since it is able to provide the throughput required for streaming high frequency DRTS simulation data, as well as cross platform and multi language support. This streaming module listens for messages published by the first module, and publishes them to the shared ZeroMQ message broker. The message payload on ZeroMQ topics is the decoded array of floating point values. There exists one ZeroMQ topic for each of the RTS network interfaces. Each topic publishes a homogeneous set of values and a fixed frequency. Metadata for topic messages are maintained in csv files which describe the message index and variable name for that value. To test, the RTS Data application implements a simple ZeroMQ subscriber that may be enabled to ensure data is available on the topic.

The final module features saving these ZeroMQ message to disk. Each message–an array of floating point values–is saved as a row in a CSV file. Once the location on disk and file name are defined, and the streaming modules are publishing messages, the archive module will subscribe and start writing to disk. Since the archive module listens to the shared message broker, and not the internal message broker, the archive module may run on any machine able to connect to the shared ZeroMQ service. In this respect, the storage module is one example of a downstream application.

IV. Visualization and Analysis

The data generated by the framework present a number of challenges in terms of visualization and analysis. In this section, we discuss the relevant issues to visualization, solutions, and possible implementations.

A. *Framework Output*

The chief challenge presented to visualizing these data is the high rates served by the system. State vectors from the relay can be updated as quickly as 5 kHz. Keeping in mind that most display devices are limited to refresh rates of 60 Hz, attempting to update a displayed value at the same rate the data is arriving is wasteful because the viewer will never see 98% of the values. Similarly, attempting to buffer and display a time series of all the data is not sufficient. Displaying a series with a window of 15 minutes (which is a common ADMS use case) would require storing and plotting 4.5 million points for a single variable.

Another issue comes from the use of multiple topics coming from the ZeroMQ server. Even if the data coming from two topics are being sampled at the same sample rate, different publishers will emit data at slightly different times. Network issues, such as packet processing latency, resend, etc, can further distort arrival times. If plotting a small number of variables, this should not pose a large issue, however, as users may wish to visualize several hundred variables, a different approach must be taken. Sporadic and sparse update patterns are not amenable most plotting libraries (which tend to require a reprocessing of all visible data if there is any change) or to graphics cards in general, which prefer batch memory updates.

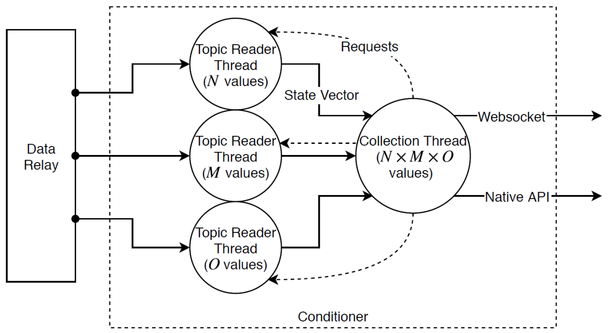
Data Relay
Topic Reader Thread (N values)
Topic Reader Thread (M values)
Topic Reader Thread (O values)
Requests
State Vector
Collection Thread ($N \times M \times O$ values)
Websocket
Native API
Conditioner

Fig. 4. Structure of the conditioner. Note that this can be used as a standalone server, or as a library for inclusion into a larger project.

To meet this challenge, a filtering step can be applied to the data and interposed between the data relay and the visualization (or analysis) tool. This tool will resample the data to a lower rate to ease the burden on visualization tooling, as well as re-framing the data to remove the issue of differing arrival times. Note that this resampling should not compromise the analysis, as the logging component of the real-time system is still active and preserves the full fidelity of the data for post-experiment analysis.

B. Conditioner

The objective of this tool is to resample the input data into a form that is simpler for the visualization tools to consume. In this function, it acts as a bridge between the ZeroMQ relay and the visualization elements by subscribing to the ZeroMQ topics, filtering/resampling, then publishing the processed data on a Websocket [19]. Websockets are a common communications protocol supported by most software platforms; ZeroMQ, though excelling as a message bus, is not as universal and requires other libraries for use, which adds a burden to analysis and visualization software development. Thus, this bridge can condition the data for easy plotting, and it can provide the data in a simple form for any number of downstream tools to use.

The conditioner is formed into several components (Fig. 4). The buffering portion subscribes to the ZeroMQ relay; each topic this is published to by the framework is monitored by a separate worker thread. In this way, we can maximize throughput, such that once off the network, no message from the relay is waiting on another. Each of these workers maintains a buffer of the last value seen for each variable that the topic publishes. Upon receipt of a message, the worker will decode the string, form the array of floating point values, reformat signal variables (that is, if a floating point variable is used to represent a boolean, the value is clamped to 0 or 1), and copy the values to the buffer.

The second portion is the collection phase, which operates by a timer, using a user-selected interval. When the timer is triggered, the collector will ask each worker for their current buffer of values. These replies are all assembled into a single large state vector. Once assembled, this vector is then packed and sent to the Websocket portion.

The Websocket side consists of a server that handles client bookkeeping and data distribution. When a new vector is ready, sending is handled asynchronously, with each client handled independently. The data is sent as a packed binary array of floating point values, eliminating the need for string decoding. The Websocket also provides a metadata mechanism to help clients decode the large binary state vector. The conditioner is written in C++ using the Qt framework [20]. While deployable as a standalone tool, the conditioner can also be quickly adapted as a library for inclusion directly into a visualization application.

C. Visualization

To provide a high-performance visualization platform, a custom 2-D plotting application was developed in C++, Qt, and OpenGL. Many plotting libraries are intended to operate on static or rarely changing data. When used out of this comfort zone with higher rate data and many plots, this is problematic.

For this application, plotting is considered to be of two types: real-time (for line plots) and staged (for waveforms). The former plot type uses the conditioner as a library for data, and the second directly reads from the ZeroMQ relay. The latter mode enables researchers to visualize the full rate of data, much like an oscilloscope, where screen updates are done at a fixed refresh, but the full number of samples can be displayed (for example, for a 1-kHz sample rate, we would display 1,000 samples updated every second).

Normal plotting can take advantage of the packed conditioner output. In this application, vertex buffers and index buffers are arranged in a circular buffer. On an update, data from the conditioner are copied to a per-chart central processing unit- (CPU-) side buffer, which can be uploaded directly into the vertex buffer with a single call, reducing CPU-to-graphics processing unit (GPU) communications overheads. Because we are merely uploading new values, the size of the upload is related to the number of lines the user wishes to plot on a single chart, not the whole sampled range over all the lines. The use of a circular buffer means that at most two draw calls to the GPU will be issued per plot.

For oscilloscope plots, a delay buffer is used, where data are accumulated for a duration of time (typically 1 second), and when the buffer is full, the data will be displayed to the user. While slowly updating, it can provide the researcher a view into the subsecond waveforms of a dynamic variable. A link to the application can be found in [21].

V. Results

Fig. 5 shows the developed application at the laboratory space dedicated for the ADMS evaluation. Two types of visualization applications were developed and tested using the framework developed in this work. Because this application was developed for control from a central location, a black background was used in the visualization tools. Fig. 5 shows the ADMS under evaluation, the DRTS used to simulate the power system, controller hardware used in the setup and the visualization tool that was developed using the proposed approach. The data generated by the DRTS was relayed and stored using the proposed application. The visualization shown in Fig. 5 is used to show the critical points of the experiments, voltage at the feeder model, and the voltage at the capacitor banks 1 and 2 controlled by the ADMS.

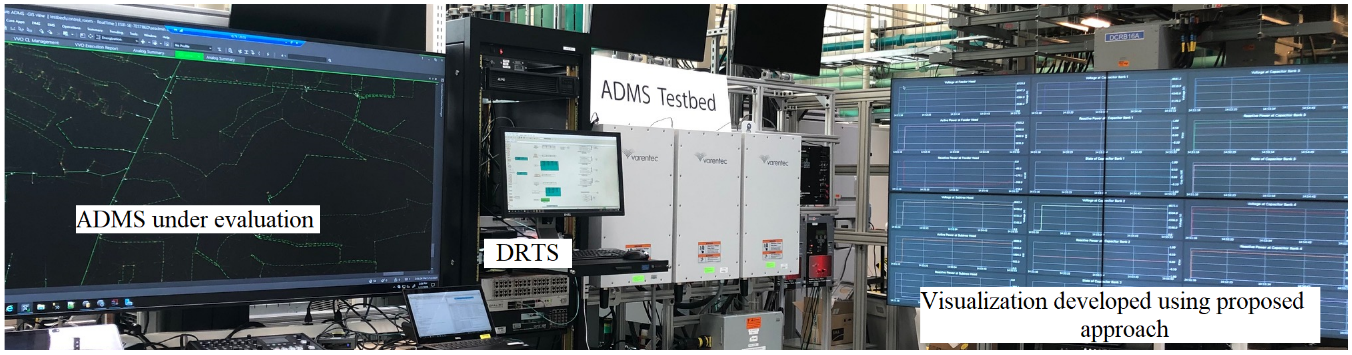

Fig. 5. Use of developed architecture and framework for visualization in control center setup

VI. Future work

In this research work, the preliminary results from open source framework was presented. In the future work, the results will be compared with existing approaches using analytical metrics. In this paper, line plots were generated to visualize the key parameters in the modeled system, this will be replaced with a heat map of the distributed system.

VII. Conclusion

Data storage and real-time visualization of CHIL/PHIL experiments can be of great value. The data storage capability provided through the application developed here will enable better insight into the models being simulated and the experiments being performed in real-time. The visualization tool developed in this research work is intended for use in a laboratory control room setup. The real-time visualization application developed is crucial for running PHIL experiments because identifying an issue in the simulation as soon as possible will save significant time and effort. The flowchart for data relay and the visualization application were presented. The two applications developed in this work have been successfully used in multiple projects. This open-source tool could be used by other researchers to replicate this approach in other possible languages.

Acknowledgment

We thank the anonymous reviewers for their valuable comments and recommendations. The authors gratefully acknowledge the contributions of Soumya Tiwari, and Ismael Mendoza to the test bed design, implementation, and running multiple experiments.

References

[1] J. C. Mankins, "Technology readiness levels," *A White Paper, NASA, Washington, DC*, 1995.

[2] ——, "Technology readiness assessments: A retrospective," *Acta Astronautica*, vol. 65, no. 9, pp. 1216–1223, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0094576509002008

[3] A. Maitra, A. Pratt, T. Hubert, D. Wang, K. Prabakar, R. Handa, M. Baggu, and M. McGranaghan, "Microgrid controllers: expanding their role and evaluating their performance," *IEEE Power and Energy Magazine*, vol. 15, no. 4, pp. 41–49, 2017.

[4] N. Ainsworth, A. Hariri, K. Prabakar, A. Pratt, and M. Baggu, "Modeling and compensation design for a power hardware-in-the-loop simulation of an AC distribution system," in *2016 North American Power Symposium (NAPS)*, 2016, pp. 1–6.

[5] W. Ren, M. Steurer, and T. L. Baldwin, "Improve the stability and the accuracy of power hardware-in-the-loop simulation by selecting appropriate interface algorithms," *IEEE Transactions on Industry Applications*, vol. 44, no. 4, pp. 1286–1294, 2008.

[6] J. Wang, J. Fossum, K. Prabakar, A. Pratt, and M. Baggu, "Development of application function blocks for power-hardware-in-the-loop testing of grid-connected inverters," in *2018 9th IEEE International Symposium on Power Electronics for Distributed Generation Systems (PEDG)*, June 2018, pp. 1–8.

[7] T. J. Overbye and J. D. Weber, "Visualization of power system data," in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, Jan 2000, pp. 7 pp.–.

[8] C. Mikkelsen, J. Johansson, and M. Cooper, "Visualization of power system data on situation overview displays," in *2012 16th International Conference on Information Visualisation*, July 2012, pp. 188–197.

[9] "Distributed messaging," http://zeromq.org/, 2018.

[10] A. Pratt, M. M. Baggu, F. Ding, S. Veda, I. Mendoza, and E. Lightner, "A test bed to evaluate advanced distribution management systems for modern power systems," in *2019 IEEE EUROCON*. IEEE, 2019, pp. 1–6.

[11] A. Singh and K. Prabakar, "Controller-hardware-in-the-loop testbed for fast-switching sic-based 50-kw pv inverter," in *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*, Oct 2018, pp. 1109–1115.

[12] M. Steurer, C. S. Edrington, M. Sloderbeck, W. Ren, and J. Langston, "A megawatt-scale power hardware-in-the-loop simulation setup for motor drives," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 4, pp. 1254–1260, 2010.

[13] P. Koralewicz, V. Gevorgian, R. Wallen, W. van der Merwe, and P. Jörg, "Advanced grid simulator for multi-megawatt power converter testing and certification," in *2016 IEEE Energy Conversion Congress and Exposition (ECCE)*. IEEE, 2016, pp. 1–8.

[14] V. Gevorgian, P. Koralewicz, R. Wallen, and E. Muljadi, "Controllable grid interface for testing ancillary service controls and fault performance of utility-scale wind power generation," National Renewable Energy Lab.(NREL), Golden, CO (United States), Tech. Rep., 2017.

[15] S. Veda, M. Baggu, and A. Pratt, "Defining a use case for adms testbed: Data quality requirements for adms deployment," in *2019 IEEE Power Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, Feb 2019, pp. 1–5.

[16] M. M. Baggu and A. Pratt, "NREL's advanced distribution management system (ADMS) test bed."

[17] A. Pratt, M. Baggu, F. Ding, S. Veda, I. Mendoza, and E. Lightner, "A test bed to evaluate advanced distribution management systems for modern power systems," in *IEEE EUROCON 2019 -18th International Conference on Smart Technologies*, pp. 1–6, ISSN: null.

[18] "Open source data handler," https://github.com/NREL/rts-data, 2018.

[19] "The websocket protocol," https://tools.ietf.org/html/rfc6455, 2018.

[20] "Qt," https://www.qt.io, 2018.

[21] "Data visualization," https://github.com/NREL/rts-vis-app, 2018.