OPQ Version 2: An Architecture for Distributed, Real-Time, High Performance Power Data Acquisition, Analysis, and Visualization

Anthony J. Christe, Sergey I. Negrashov, Philip M. Johnson, Dylan Nakahodo, David Badke, David Aghalarpour
Collaborative Software Development Lab
Department of Information and Computer Sciences
University of Hawaii at Manoa

***Abstract*—OpenPowerQuality (OPQ) is a framework that supports end-to-end capture, analysis, and visualizations of distributed real-time power quality (PQ) data. Version 2 of OPQ builds on version 1 by providing higher sampling rates, optional battery backup, end-to-end security, GPS synchronization, pluggable analysis, and a real-time visualization framework. OPQ provides real-time distributed power measurements which allows users to see both local PQ events and grid-wide PQ events. The OPQ project has three principal components: back-end hardware for making power measurements, middleware for data acquisition and analysis, and a front-end providing visualizations. OPQBox2 is a hardware platform that takes PQ measurements, provides onboard analysis, and securely transfers data to our middleware. The OPQ middleware performs filtering on the OPQBox2 sensor data and performs high-level PQ analysis. The results of our PQ analysis and real-time state of the sensor network are displayed using OPQView. We've collected distributed PQ data from locations across Oahu, Hawaii and have demonstrated our ability to detect both local and grid-wide power quality events.**

I. Introduction

As power grids transition from a centralized generation model to a distributed model, maintaining stability requires fine grained knowledge of the grid's state[1]. Monitoring power quality (PQ) on a distributed generation smartgrid requires a distributed sensor network. How do we collect, analyze, and visualize PQ at the power grid scale using data gathered from residential utility customers?

To answer this question, we developed and deployed an open source hardware and software system focused on consumer level monitoring across Oahu called Open Power Quality (OPQ). Version 2 of OPQ (OPQ2) is able to aggregate distributed PQ measurements, perform high level real-time PQ analysis and classification, and display PQ at both the consumer and grid level. OPQ2 is a multi-layered framework consisting of custom hardware for collecting distributed PQ data (OPQBox2), a middleware for filtering and event detection (OPQMakai), higher level PQ analysis (OPQMauka), and finally a front-end for consumer friendly grid level PQ visualizations (OPQView). The OPQ2 architecture is shown in figure I.

Each OPQBox2 processes 2.7 billion samples per day. Using node level feature extraction, OPQBox2's forwards only 500,000 measurements to the cloud service in the same 24 hours. Further analysis and filtering yields roughly 1 distributed PQ event per day. These events contain data from every device on the network sampled at full resolution and sampling rate. Through the use of our multi-tiered system, we aim to bring grid operators and end users, useful, intuitive, actionable, real-time information about the power grid.

II. OPQ Version 2 Architecture

A. OPQBox2

OPQBox2 is a modern, high performance, power quality monitor built with distributed measurements in mind. OPQBox2 provides high resolution sampled data of up to 50kS/s at 16bits. Optional GPS synchronization and battery backup can be added to the OPQBox2 to tailor it to a specific power quality measurement.

While sampling and GPS synchronization is controlled by the realtime DSP, all of the signal processing and communication is controlled by a Raspberry PI single board computer. The Raspberry PI collects 10 AC cycles of ADC measurements at a time and computes the utility frequency and V_{rms}. These values are sent to the cloud triggering broker via WiFi while the raw waveforms are buffered in the local Redis key value store.

If the triggering data stream indicates a power quality event, our middleware may request raw waveforms from the affected OPQBox2s. When the triggering server requests data from an OpqBox2, it requests a time range. This data is queried from Redis, serialized into a single large data packet, and transferred back to the requesting server. All of the communication between the OPQBox2 and the cloud services are done over an encrypted ZeroMQ connection[2].

Time synchronization of the OPQBox2 is performed via Network Time Protocol (NTP)[3]. While OPQBox2's are able to synchronize via GPS, we found NTP to be suitable for our system. We have verified the synchronization provided by Network Time Protocol (NTP) by comparing frequency measurements collected from two OPQBox2 devices and examining their phase difference.

B. OPQHub

OPQHub is OPQ's middleware system and is responsible for collecting raw and feature extracted data from OPQBox2's and for performing high level PQ analysis. OPQHub is split into

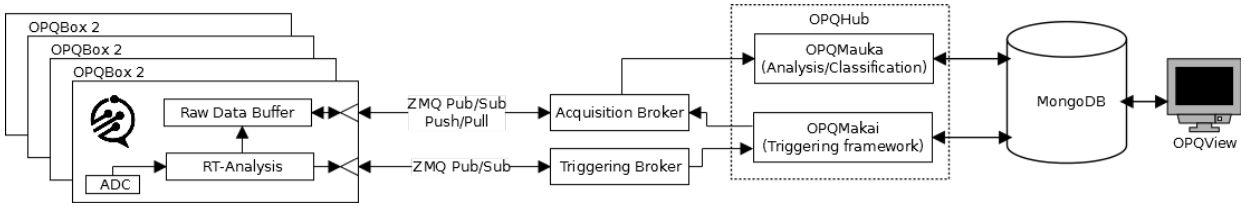

Fig. 1. OPQ System Architecture

two components, OPQMakai, a pluggable filtering and data request component and OPQMauka, a pluggable real-time PQ analysis component.

OPQHub is a drastic change from OPQv1. OPQv1 performed all collection, analysis, and display from OPQBoxes on a single server. OPQv1 was incapable of scaling to larger data loads and did not display a clear separation of concerns. Our second iteration of software updates this by providing separate, distributed layers for both triggering of raw PQ data and higher level PQ analysis.

All communication between the OPQBox2 and OPQHub passes through two connection brokers. These brokers allow us to terminate encryption at the cloud boundary, thus simplifying the analysis framework. ZeroMQ is used throughout our cloud infrastructure to connect the analysis plugins to brokers and other plugins. This allows us to form ad-hoc analysis topologies, such pipeline processing, and map reduce pipelines without interfering with regular data acquisition.

The triggering stream consists of the feature reduced data(frequency and V_{rms}) for each device in the OPQ2 network. This stream is brokered via the triggering broker, and analyzed by the set of analysis plugins called OPQMakai. If multiple devices show temporally coherent deviation from the norm, OPQMakai will request the raw data from OPQBox2 devices via the acquisition broker.

Further analysis of the raw waveform is performed by the OPQMauka analysis and classification system. Each of the plugins in OPQMauka are responsible for different PQ classification and analysis tasks. We currently implement plugins that detect and report PQ measurements, voltage sags and swells, frequency sags and swells, and perform ITIC event classification of voltage events. We've also developed plugins that, along with the filtering framework OPQMakai, allow us to detect these events on a distributed level. Data products from OPQMauka are stored in MongoDB and presented to clients use OPQView.

OPQMakai and OPQMauaka are described in further detail in the following sections.

C. OPQMakai

D. OPQMauka

OPQMauka is a distributed high-level sensor data analysis platform. OPQMauka provides a distributed computation model that is capable of scaling with the amount of sensor data it is receiving. The OPQMauka processing pipeline is implemented as a directed acyclic graph (DAG). Communication

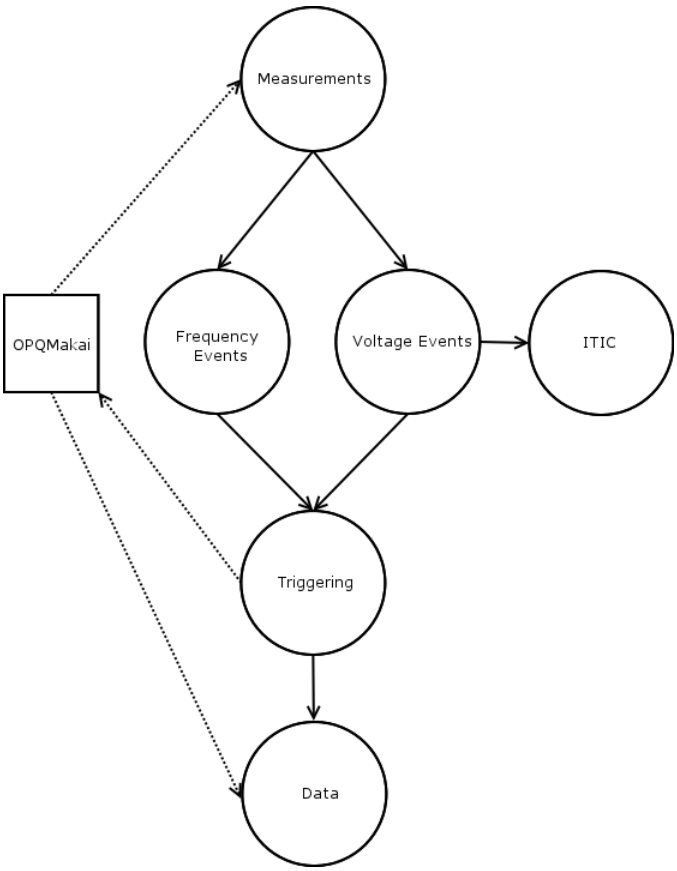

Fig. 2. Overview of OPQMauka Architecture as a DAG

between vertices in the graph is provided via ZeroMQ. Each node in the graph is implemented by an OPQMauka Plugin.

Each OPQMauka Plugin provides a set of topics that it subscribes to and a set of topics that it produces. These topics form the edges between vertices in our graph. Because each plugin is independent and only relies on retrieving and transmitting data over ZeroMQ, plugins can be implemented in any programming language and executed on any machine in a network. This design allows us to easily scale plugins across multiple machines in order to increase throughput. Further, this system makes it possible to create and activate new plugins into the OPQMauka framework. Plugins can be enabled or disabled without affected the rest of the framework. The structure of OPQMauka is show in figure 2.

The following paragraphs briefly describe the working of

each plugin within the OPQMauka framework.

The OPQMauka Measurements Plugin subscribes to OPQMakai's triggering data stream, and samples the data at a configurable interval and stores these measurements in MongoDB for providing realtime visualizations of PQ data. This plugin produces measurement messages which are used by the voltage and frequency plugins to determine high-level voltage and frequency PQ events.

The OPQMauka Voltage Plugin subscribes to measurement messages and identifies voltage swells and voltage sags. The threshold by which the plugin identifies swells and sags is configurable. For the results of this paper, we set the threshold to be +/- 5% of 120 Volts. Once voltage events are identified, the event is stored in a MongoDB and a voltage event message is produced.

The OPQMauka Frequency Plugin subscribes to measurement messages and identifies frequency swells and frequency sags. The threshold by which the plugin identifies swells and sags is configurable. For the results of this paper, we set the threshold to be +/- 1% of 60 Hz. Once frequency events are identified, the event is stored in a MongoDB and a frequency event message is produced.

The OPQMauka ITIC Plugin subscribes to voltage events and classifies them using the ITIC Curve[4], a power standardization curve that determines the likelihood of causing damage to consumer equipment based on the magnitude and duration of a voltage event. Classified ITIC events are stored in MongoDB.

The OPQMauka Triggering Plugin subscribes to voltage and frequency event messages, and requests raw PQ data from the network. By monitoring high-level voltage and frequency events, we can selectively decide which sets of raw PQ data should be analyzed for transient detection. This plugin interfaces and works alongside OPQMakai by utilizing a request/response pattern over a ZeroMQ socket. When this plugin requests data from OPQMakai, OPQMakai immediately responds with an *eventid*. This plugin will then forward the *eventid* to the OPQMauka Data Plugin which will asynchronously receive raw data payloads from OPQMakai.

The OPQMauka Data Plugin subscribes to acquisition messages, waits a configurable amount of time for raw data from OPQMakai to be populated from Redis, and then reads the data from Redis, deserializes the raw data, and stores the data and associated meta-data in MongoDB for further DSP processing and historical analysis.

Finally, the OPQMauka Status Plugin subscribes to heartbeat messages which are produced by all OPQMauka plugins. Heartbeat messages contain optional statistics about the state of individual plugins and their respective workloads. The status plugin stores health information in MongoDB which can be queried to examine the overall health of OPQMauka and its distributed plugins.

Table I summarizes the topics that each plugin subscribes and publishes to.

Plugin	Subscribes	Publishes
Measurement	Triggering messages	Heartbeat, Measurement
Voltage	Measurement	Heartbeat, VoltageEvt
Frequency	Measurement	Heartbeat, FrequencyEvt
ITIC	VoltageEvt	Heartbeat, ITIC
Triggering	FrequencyEvt, VoltageEvt	Heartbeat, Acquisition
Data	Acquisition	Heartbeat, DataStored
Status	Heartbeat	

TABLE I
OPQMAUKA PLUGINS: PUB/SUB OVERVIEW

E. OPQView

OPQView serves as the front-end user interface of the OPQ2 system. It is implemented in JavaScript using MeteorJS. OPQView serves as the end-point of OPQ2. OPQView is responsible for displaying collected PQ data in a useful and meaningful way. OPQView maintains a communications interface with OPQHub via a MongoDB database, from which it is able to reactively retrieve real-time PQ data. This data includes voltages and frequencies measured by individual devices, as well as PQ events and their corresponding waveforms. OPQView does not only serve as a front-end to display real-time data, but it can also display historical grid trends and power quality events.

The real-time nature of our data allows for a wide potential of visualization techniques - such as real-time heat maps of device voltages and frequencies across the grid. OPQView also supplies the administrative interface for OPQBox2's, allowing device owners to adjust their device's privacy and sharing settings as needed.

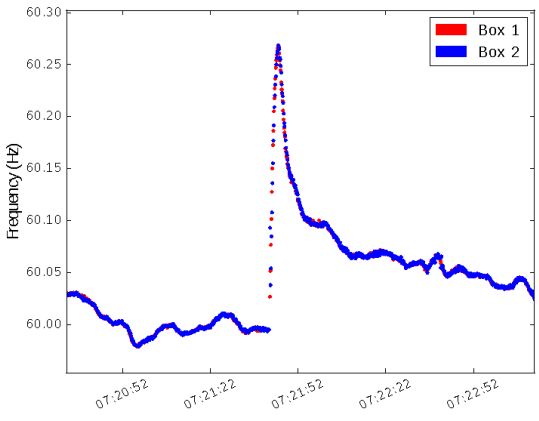

Fig. 3. Frequency event. March 1st 2017.

III. RESULTS

Currently, OPQHub is capable of detecting and recording frequency and voltage based PQ events. Voltage based events include sags, swells. Frequency events are limited to frequency deviation from the 60Hz nominal. Example of a triggering stream of a frequency event is shown in Figure 3. This event occurred during the lighting storm March 1st 2017, and is

likely a lighting strike. Two devices were separated by 5 miles and were connected to different substations.

Frequency deviations on the order of $\frac{1}{4}$Hz, can lead to load shedding[5]. This is a process where a section of the power grid is disconnected from the utility in order to preserve grid stability. Collection and analysis of events such as this one will ultimately lead to insights into power grid operation, and give utility customers a more active role in power grid monitoring.

REFERENCES

[1] T. Peffer, "Evaluation of challenges and potential applications of building-to-grid implementation," *California Institute for Energy and the Environment Whitepaper*, Oct. 2010.

[2] P. Fengping and C. Jianzheng, "Distributed system based on zeromq," *Electronic Test*, vol. 7, no. 7, pp. 24–29, 2012.

[3] D. L. Mills, "Internet time synchronization: the network time protocol," *IEEE Transactions on communications*, vol. 39, no. 10, pp. 1482–1493, 1991.

[4] "Power quality in electrical systems," Apr 2011. [Online]. Available: http://www.powerqualityworld.com/2011/04/itic-power-acceptability-curve.html

[5] W. C. New, "Load shedding, load restoration and generator protection using solid-state and electromechanical underfrequency relays," *General Electrics White Paper*, 2014.