Spring 2024

Development of a Collaborative Research Platform for Efficient Data Management and Visualization of Qubit Control

Devanshu Brahmbhatt

Development of a Collaborative Research Platform for Efficient Data Management and Visualization of Qubit Control

by

DEVANSHU BRAHMBHATT

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON
May 2024

ACKNOWLEDGEMENTS

I extend my deepest gratitude to my supervising professor, Dr. VP Nguyen, for his continuous guidance and encouragement throughout my graduate studies. His profound expertise in various sub-domains of computer science and broad knowledge of the field have been invaluable to my research journey. I am also thankful to my committee members, Dr. Mohammad Atiqul Islam and Dr. Yilun Xu, for their keen interest in my work and for dedicating their time to serve on my committee. I am especially appreciative of the stress-free research environment that Dr. Nguyen fostered, allowing me to explore a wide array of topics that piqued my interest. My thanks also go to all the professors who have taught and inspired me during my time as a graduate student. I would like to express my heartfelt appreciation to my lab partners and colleagues for their collaboration and the many ways in which they have enriched my research experience. Their camaraderie and support have made my journey through graduate school not only productive but also enjoyable. To my friends, who have provided a network of support and relief from the stresses of academic life, thank you for all the moments of fun, creativity, and relaxation. Your friendship has been a treasured part of my life during these years. Lastly, I am profoundly grateful for the opportunity to conduct my research at one of the world's leading institutions, surrounded by some of the brightest minds in the field. This experience has not only allowed me to grow intellectually but also to think critically about my career path ahead. Thank you for this remarkable opportunity and for believing in my potential. I also extend my heartfelt thanks to my parents for their unwavering support and encouragement, and for financially supporting my graduate education. Their belief in my abilities has been a constant source of motivation

April 26^{th}, 2024

ABSTRACT

Development of a Collaborative Research Platform for Efficient Data Management and Visualization of Qubit Control

Devanshu Brahmbhatt, MS

The University of Texas at Arlington, 2024

Supervising Professor: Dr. Phuc VP Nguyen

This thesis introduces QubiCSV, a pioneering open-source platform for quantum computing field. With an emphasis on collaborative research, QubiCSV addresses the critical need for specialized data management and visualization tools in qubit control. The platform is crafted to overcome the challenges posed by the high costs and complexities associated with quantum experimental setups. It emphasizes efficient utilization of resources through shared ideas, data, and implementation strategies. One of the primary obstacles in quantum computing research has been the ineffective management of extensive calibration data and the inability to visualize complex quantum experiment outcomes effectively. QubiCSV fills this gap by offering robust data versioning capabilities for storing calibration and characterization data, essential in qubit control systems. Additionally, it enables researchers and programmers to interact with qubits in real time, enhancing their understanding and optimizing qubit performance. The platform's advanced visualization tools are specifically tailored to interpret intricate quantum experiments, facilitating deeper insights into qubit behavior. Overall, QubiCSV stands as a significant advancement in quan-

tum computing research, streamlining data handling and elevating user experience with intuitive visualization, thereby becoming an invaluable resource for researchers in this domain.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

CHAPTER 1

Introduction to Quantum Computing and Qubit Control

Quantum computing represents a significant evolution from classical computing, harnessing the principles of quantum mechanics to process information. Classical computers use bits as the basic unit of data, which exist in one of two states: 0 or 1. In contrast, quantum computers use quantum bits, or qubits, which can exist in multiple states simultaneously due to quantum superposition. This capability allows quantum computers to solve certain problems — such as those involving integer factorization or quantum simulation — much more quickly than classical computers. For instance, Google's quantum computer, Sycamore, demonstrated this by performing a complex computation in 200 seconds that would take the fastest supercomputer about 10,000 years. In quantum computers, quantum operations are applied to quantum registers that contain qubits. Unlike classical bits, qubits engage in phenomena like entanglement and superposition, which are the cornerstones of quantum computation. The architectural difference between quantum and classical systems is profound, with quantum systems utilizing qubits instead of bits, resulting in fundamentally different computational approaches

1.1 Quantum Computing vs. Classical Computing

The transformative potential of quantum computing contrasts starkly with traditional classical computing, primarily due to the fundamental differences in their computational units and mechanisms. Classical computers utilize bits as the smallest unit of data, which can either be 0 or 1. In contrast, quantum computers use quan-

tum bits or qubits, which can exist simultaneously in multiple states beyond binary, thanks to quantum superposition.

This unique capability allows quantum computers to solve certain complex problems, which are infeasible for classical computers, in a drastically reduced time frame. For instance, quantum computers can perform calculations in parallel due to entanglement and superposition, providing an exponential speed-up for problems like integer factorization and quantum simulation. This was notably demonstrated by Google's 53-qubit quantum computer, Sycamore, which solved a problem in minutes that would take thousands of years for the best classical computers

1.2 Qubits in Quantum Computing

Introduction to Qubits: In a classical computer, information is represented by bits, which can be either 0 or 1. Computation is performed using logic gates that manipulate these bits. The state of a classical computer is determined by the states of all its bits, allowing it to exist in one of 2^n possible states, where n is the number of bits [1]. However, in quantum computing, the basic unit of information is called a qubit. Unlike classical bits, qubits can exist in a superposition of states. This means that a qubit can represent not only 0 or 1 but also a combination of both simultaneously. For example, a qubit can be in a state that is 30 Percentage "0" and 70 Percentage "1" at the same time [2]. This property of superposition gives quantum computers a much richer set of states compared to classical computers.

Quantum Entanglement: Furthermore, qubits can exhibit another phenomenon called quantum entanglement. Entanglement is a unique property where the states of two or more qubits become correlated, even when they are physically separated. This correlation allows qubits to share information instantaneously, regardless of the distance between them. Quantum computers harness the power of entanglement and

superposition to perform computations. Instead of being limited to a single state at a time, qubits in a quantum computer can represent an exponentially large number of states simultaneously. This enables quantum computers to process information in parallel, potentially solving certain problems much faster than classical computers.

1.2.1 Superconducting qubits

Introduction to Superconducting Qubits: Superconducting is prominent technology in quantum space, Major players in Industry has chosen superconducting qubits to design quantum processor[3]. it holds reliable scalable architecture, IBM and China have achieved the highest scalability with 433 qubits and 121 qubits respectively[4], but the qubit quality is not yet suitable for practical use.[5]

Quantum Operations in Superconducting Qubits: Superconducting qubits operate based on the principles of quantum superposition and quantum coherence. They utilize Josephson junctions, which are thin insulating barriers between superconducting materials, to create quantum electrical components.The behavior of superconducting qubits is governed by the manipulation of the superconducting phase difference or the direction of superconducting current flow in the qubit circuit.[6]
One common approach is microwave pulses, which are used to manipulate the qubit's quantum state by driving transitions between energy levels. These transitions correspond to operations such as single-qubit gates (e.g., rotations) and two-qubit gates (e.g., entanglement operations).

Challenges in Superconducting Qubits: implementing quantum operations in superconducting qubits faces challenges. One challenge is qubit decoherence[7], which refers to the loss of quantum information due to interactions with the surrounding environment. Decoherence sources include intrinsic noise from the Josephson junc-

tion itself, as well as external electromagnetic perturbations. Efforts are made to improve qubit coherence by minimizing environmental interactions and optimizing the qubit design to reduce noise sources. Additionally, achieving high-fidelity gate operations and maintaining long coherence times are ongoing challenges in superconducting qubits [8]. Improvements in qubit design, material quality, and control techniques are pursued to address these challenges and enhance the performance of superconducting qubits.

1.3 Introduction: Qubit Control

Qubit control is the process of manipulating the state of a qubit, a quantum mechanical system that can exist in a superposition of two states. This can be done by applying a series of RF pulses to the qubit, which are generated by an electronic system. The electronic system consists of a field-programmable gate array (FPGA), which is a chip that can be programmed to perform complex operations. The FPGA generates the RF pulses according to a predetermined sequence, which is specified by a quantum programming language. The RF pulses are then applied to the qubit, which causes it to change its state. The state of the qubit can then be read out by measuring the qubit's output signal.[9] Controlling qubits effectively is pivotal for the advancement of quantum technologies [10, 11]. Qubit control involves precise manipulation to perform quantum operations, which is challenging due to their susceptibility to noise and decoherence [12, 13]. Managing these challenges is critical, as the sensitivity of qubits to their environment can greatly impact the accuracy and reliability of quantum operations. The need for robust control systems is therefore essential for advancing towards fully fault-tolerant quantum computing.

CHAPTER 2

Qubit Control Systems

2.1 Introduction

Qubit control [10], the manipulation of a qubit is typically executed through RF pulses generated by an electronic system, and FPGA-based solution has been used to produce RF pulses that alter the qubit's state, subsequently measured for output [9]. Several state-of-the-art systems in the qubit control landscape include QCSS Zurich Instruments [14], Qblox [15], Keysight [16], and Quantum Machines [17], each contributing to the field's progression. Among these, The QubiC (Qubit Control) [18, 19] system, developed at Lawrence Berkeley National Laboratory (LBNL), represents a notable open-source advancement. QubiC is an FPGA-based control and measurement system, specifically tailored for superconducting [6] quantum information processors. Its open-source nature, high performance, and modular design align it closely with the needs of quantum research, positioning it as a valuable tool for scientists in navigating the evolving quantum computing domain.

FPGA-based qubit control is a crucial aspect of superconducting quantum computing systems. It involves generating and routing complex sequences of radio frequency (RF) signals from room temperature electronics to the quantum processor at cryogenic temperatures. To address the challenges posed by the increasing complexity of quantum systems, a modular design approach has been adopted. The system is divided into different functions, such as timing control, arbitrary waveform generation, data acquisition, and bias voltage generation, which are implemented as separate modules interconnected through a high-speed backplane. Each module is based on

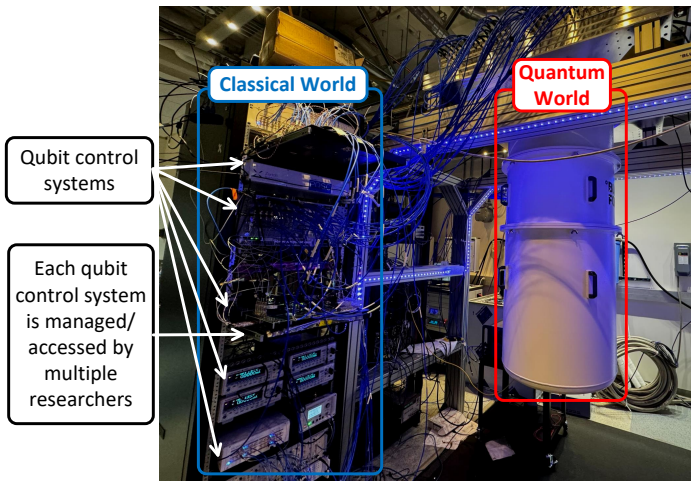

Figure 2.1. An example setting of quantum control system research.

an FPGA, allowing for specific functions while maintaining compatibility with the backplane interface.

Components of Qubit Control Systems: Qubit control systems rely on a combination of hardware components, including analog-to-digital converters (ADCs), field-programmable gate arrays (FPGAs), and software interfaces. The electronic system for qubit control must be able to generate control pulses with very high precision and timing. This is because the state of a qubit can be very fragile and can be easily changed by noise. The electronic system must also be able to generate a large number of control pulses very quickly. This is because qubits can decohere, or lose their quantum state, very quickly[20].

2.2 Qubit Control Challanges

2.2.1 Maintaining Quantum Properties and Coherence:

One of the major challenges in superconducting qubits is whether can qubits maintain their quantum properties and be accurately controlled, and how well they

are isolated enough to maintain long coherence times and achieve high-fidelity control of their quantum state, [21] [11]

2.2.2 Shrinking the Quantum Computer Footprint:

Quantum computers currently resemble large, distributed systems occupying significant physical space. As they scale up, it becomes essential to dramatically shrink their footprint, similar to how classical computers evolved from room-sized machines to compact integrated devices. The size of the system presents challenges in signal propagation, especially at radio or microwave frequencies used for quantum control signals. Working with controlled impedance and managing transmission lines becomes more complex, and differences in path length between these lines introduce significant phase shifts, particularly when synchronizing broadband signals. Moreover, there is a consideration of time-of-flight latency for signals to travel between room-temperature electronics and qubits operating at deep cryogenic temperatures, which can affect precise triggering and synchronization in a large-scale multi-qubit system.[18]

2.2.3 Unique Challenges in Scaling Qubits:

When it comes to scaling up qubits in quantum computers, there are distinct challenges that differ from those faced in classical computing. In classical systems, managing data flow between different components is relatively straightforward, but with large arrays of qubits, it becomes more complex. Unlike classical components, qubits cannot be separated from their classical control interface or treated as independent "black boxes." They require special attention due to their delicate nature and the demanding conditions in which they operate. This means that unique approaches and solutions are needed to address the specific challenges posed by qubits, taking

into account their fragility and the constraints imposed by their extreme operating environments.

2.3 QubiC: Open-source Qubit Control Designed by Berkeley Lab

QubiC is an open-source FPGA-based control and measurement system designed for quantum information processing. It provides a modular and flexible framework for implementing digital tune-up sequences and algorithmic protocols at the lowest level of qubit control. The gateware programmed on the FPGA consists of three modules: the board support package (BSP), the DSP (digital signal processing), and the host interface (HOI). The BSP handles low-level hardware configuration, while the DSP module enables basic qubit control and measurement functions independent of specific FPGA/ADC/DAC selection. The HOI serves as the interface between the host computer and the FPGA, facilitating input/output communication. QubiC's FPGA gateware enables waveform generation, pulse sequence reuse, fast qubit reset, and efficient integration of control and measurement operations, contributing to advancements in quantum computing research and applications.It consist of 3 Major Module, 1) Hardware, 2) Gateware and 3) Software

2.3.1 Hardware

FPGA/ADC/DAC Module: This module utilizes the Xilinx Virtex-7 FPGA and Abaco Systems FMC120 boards, which provide computational capability and precision analog performance. The FPGA handles digital signal processing, while the ADC and DAC modules convert analog signals to digital and vice versa. **RF Mixing Module:** This module integrates the in-phase / quadrature (I/Q) mixer, power-level adjustments, and dc bias fine-tuning. It converts the signal frequency to/from the target frequency and works with RF and local oscillator (LO) frequencies between

2.5 and 8.5 GHz. **LO Generation Module**: This module generates the LO signals required for qubit operation. It uses multiple phase-locked loops (PLL) with a shared master oscillator (MO) to generate adjustable LO frequencies. The module uses ultralow-noise crystal oscillators and evaluation modules for PLL.

2.3.2 Gateware

The synchronization between modules is achieved through three layers of protocols: 1. JESD204B subclass 1 is used for synchronizing multiple DAC chips. 2. General-purpose input/output triggers handle event information with low latency among modules. 3. Fiber-based synchronization similar to the white rabbit system is under development to lock clock phases among modules and provide high-speed data communication. The gateway DSP module is responsible for generating pulses at a specified carrier frequency with arbitrary amplitude modulation and detecting the pulse after it passes through the readout resonator. It includes processing elements for digital up- or downconversion between the IF and baseband signals. The processing elements can be configured as upconverters or downconverters, generating in-phase and quadrature-phase components. 1. **Board Support Package (BSP):** Handles low-level hardware configuration and initialization. 2. **DSP (Digital Signal Processing):** Implements basic qubit control and measurement functions independently of the FPGA/ADC/DAC selection. It runs in a single clock domain, and the HOI and BSP handle clock domain crossing. 3. **HOI (Host Interface):** Handles input/output to/from the host computer, serving as the interface for higher-level software. It enables waveform generation based on parameters and facilitates pulse sequence reuse.

2.3.3 Software

The API, written in Python, serves as the software interface between the quantum processor and higher-level software or algorithms. It compiles the quantum processor gate pulse specification and circuit description into gateware commands and pulse envelopes. The API supports gate optimization and circuit execution, with two types of compilers: the OPTM compiler for comprehensive chip calibration and gate optimization, and the RUNC compiler for running quantum circuits efficiently. The API also allows for data exchange with other frameworks like TrueQ, Qiskit, OpenQASM, and Cirq.

The QCVV experimental scripts are used to characterize and optimize the quantum information processor. These scripts perform tasks such as time alignment, single-tone experiments, punch-out experiments, two-tone spectroscopy, Rabi oscillations, readout correction, gate repeatability optimization, coherence time measurements, AllXY experiments, randomized benchmarking, cross resonance optimization, and two-qubit process fidelity measurements. These experiments help calibrate the gates and evaluate the performance of the quantum processor.

The GUI, implemented as a web-based client-server system, provides a user-friendly interface for controlling the quantum hardware remotely. It allows real-time adjustments of qubit bias points, initiation and termination of measurements, and display of results. The GUI server, running on a Linux machine, saves configuration and measurement data in YAML and HDF5 formats, respectively, and handles post-processing and visualization of the data.

CHAPTER 3

QubiCSV

3.1 Introduction

Quantum information processors require frequent calibration and characterization to mitigate errors arising from environmental fluctuations and imperfections in hardware, ensuring reliable and accurate computation [22]. Calibration [23] and characterization data storage are critical components in quantum experimentation and research. The precise calibration of quantum devices ensures the accuracy and reliability of QubiC and manipulation. Characterization data storage, on the other hand, provides a repository for characterization data, enabling scientists to track the performance and stability of quantum systems over time. This data is invaluable for ongoing research, allowing for the analysis of trends, the identification of anomalies, and the refinement of quantum models and algorithms.

In state-of-the-art superconducting quantum computing research, the complexity of managing extensive amounts of data poses a significant challenge. Research teams consisting of scientists and researchers work on various components such as controllers, chips, and boards, each integral to the functionality of the quantum computing system. A key element in this process is the calibration file – a comprehensive configuration file that contains critical values for all qubits and gates, for example, within the QubiC system. These calibration files are not only extensive but also dynamic, undergoing frequent updates to reflect changes in system configurations and improvements in qubit performance. Furthermore, an obstacle in this environment is the lack of a structured collaborative platform. This absence restricts the ability of

multiple physicists to seamlessly collaborate and contribute to each other's work, an essential factor in advancing quantum computing research and development.

The process of maintaining and updating these calibration files is a daunting task for scientists. Each team member, focusing on different aspects of the quantum system, generates their own calibration files, which need to be accurately tracked and updated. This task becomes increasingly challenging as the frequency of calibration increases. Moreover, the need for collaboration among scientists adds another layer of complexity. Sharing these extensive files and ensuring that all team members have access to the latest versions is a logistical hurdle, slowing efficient collaboration and progress.

Post-experimentation, the QubiC system generates an experiment result file, typically named *chip_name.data.json* This file, holds big potential for providing insights into the experiment's outcomes. However, the absence of a dedicated storage solution and a method to directly save this file from the hardware limits its utility. Scientists are left with a wealth of data but lack the means to store, manage, and analyze it effectively. in Figure 3.1, once the calibration configurations for QubiC are set up, QubiC sends signals to the quantum fridge, and after the experiment is completed, it generates the characterization.json file, which is crucial for evaluating experimental results.

To tackle the hard challenges restraining the pace of quantum computing research, there is an absolute need for a robust data management system with visualization capabilities. This system must be adept at handling the complex and demanding aspects of quantum computing data, containing both current and historical calibration and experiment data.

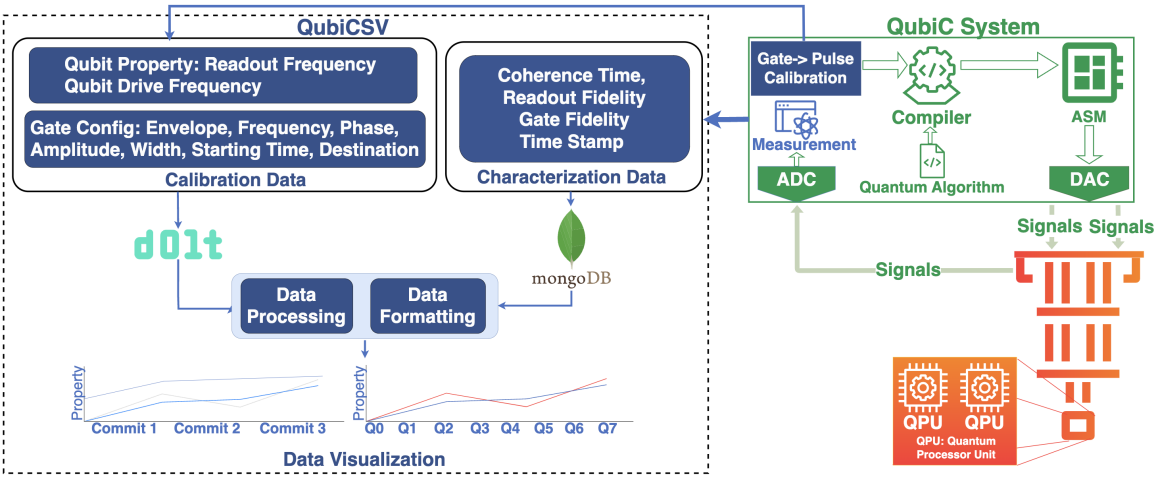

Figure 3.1. QubiCSV's Concept: System design and overview of the Data Management and Visualization System, The process begins with setting up the calibration configuration for the QubiC system, followed by the quantum algorithm and its calibration being processed through an Assembler (ASM). This information is then converted into RF signals by a Digital-to-Analog Converter (DAC) and sent to a Quantum fridge containing chips with superconducting qubits. Once the experiment is completed, the results are captured by an Analog-to-Digital Converter (ADC) and stored in a MongoDB database for visualization. If the calibration proves satisfactory for the experiments, researchers can then store the calibration file in their desired branch on the Dolt database, making it accessible for further visualization and analysis.

3.1.1 QubiCSV Motivation

we designed and implemented QubiCSV, an end-to-end platform for real-time data management and visualization for superconducting Qubit. We design our parameters to be compatible with QubiC system [19, 18], but configuration can be modified at the software level to allow the system to be compatible with other qubit control systems. The overall concept of QubiCSV is illustrated in Figure 3.2. The process starts with calibrating the QubiC system, followed by processing the quantum algorithm and calibration using an Assembler (ASM). The data is converted into RF signals by a Digital-to-Analog Converter (DAC) and sent to a dilution refrigerator with superconducting qubits. After completing the experiment, results are captured by an

Analog-to-Digital Converter (ADC) and stored in a MongoDB database for visualization. If calibration meets standards, researchers save the file in their Dolt database branch for further analysis. QubiCSV's unique approach to data storage—utilizing data versioning similar to Git but implemented in a database—provides a novel way to manage quantum data. This method allows for more effective tracking and reverting of changes over time, which is especially beneficial in a field where experiments are frequently adjusted and iterated upon. QubiCSV offers a more holistic solution to quantum computing researchers.

QubiCSV also addresses these key challenges in quantum computing with sophisticated visualization and data management capabilities. For demonstration, we implement QubiCSV to aid the QubiC team at LBNL in handling extensive and evolving calibration data, a crucial component in QubiC systems. The platform eases collaboration by simplifying sharing and updating calibration files. Post-experiment, it effectively manages experiment result files, overcoming the limitations of traditional storage methods. QubiCSV's visualization tools streamline calibration processes, reducing time and error. Designed for scalability and user-friendliness, it accommodates increasing qubits and gates, supported by a detailed user manual for ease of use. With an API response time under 500ms and a flexible MVC architecture, QubiCSV not only improves current quantum research workflows but also adapts to future needs, allowing users to customize their data management and visualization.

3.1.2 QubiCSV Data Management

QubiCSV design is inspired by Model-View-Controller (MVC) architecture [24, 25], with the model managing database queries and returning data as requested by the controller [26]. The view then renders this data, presenting it to the users in an accessible format. This system is built on the team members' daily routines, with a

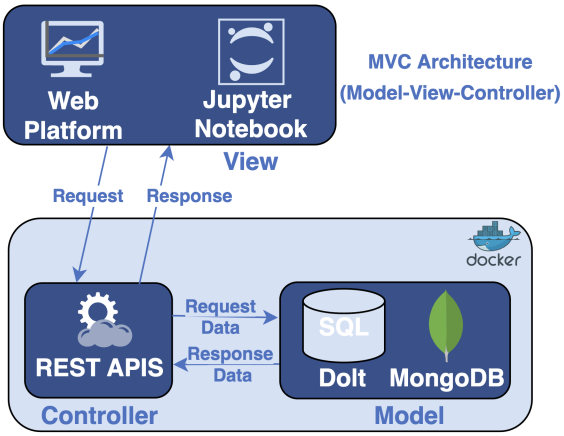

Figure 3.2. QubiCSV's system architecture..

focus on their utilization of the Jupyter Notebook for organizing code and data files, including calibration and characterization data. The combination of Dolt for calibration data and MongoDB for characterization data emerged as the best-fit solution in our design exploration. Our approach ensures a customized and effective solution tailored to the team's specific challenges. QubiCSV is designed to address both data management and visualization aspects of quantum research, mainly focusing on calibration and characterization data.

The system is structured into three major components: (1) **Application Programming Interface (API)**: It acts as the communication bridge between the database and the user interfaces, overseeing the transfer and retrieval of data to ensure smooth interaction among the system's different components [27]. (2) **Web Platform**: A user-friendly web application serves as the primary interface for users, facilitating various tasks and actions such as data management and visualization. (3) **Python Library**: Recognizing the team's reliance on Jupyter Notebook, we incorporated a Python library interface for seamless data storage and retrieval directly within their existing Jupyter Notebook workflows, as illustrated in Figure 3.2.

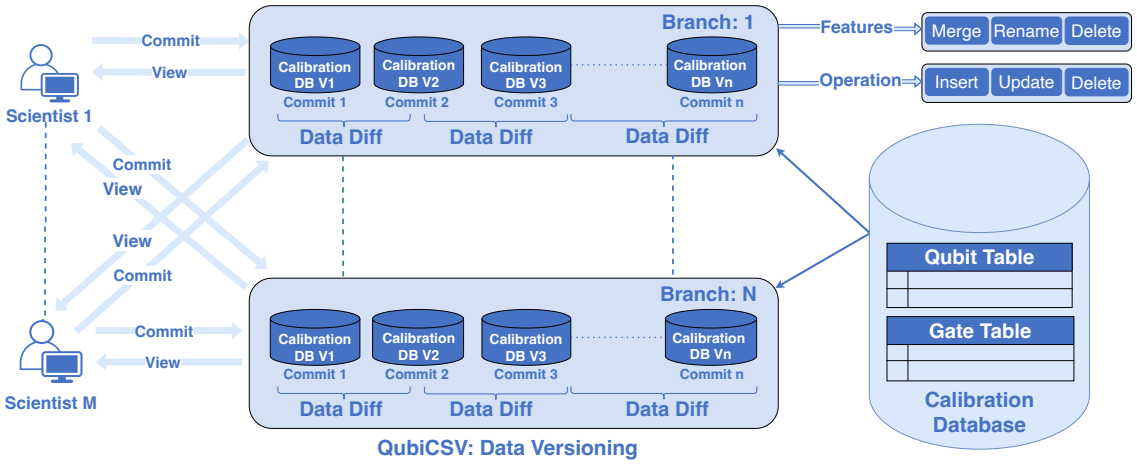

Figure 3.3. QubiCSV's data storage system with versioning capacity: This component of the system utilizes dolt, a data versioning database, enabling users to create and manage their own versions of databases in branches, similar to how git functions for source code. users can effortlessly create, access, and switch between different branches.

3.1.3 Calibration Data Management: Motivations and Approaches

Our platform leverages the data versioning database for calibration files, which is designed to address a few key needs:

(1) **Robust Tracking and Versioning:** Given the regular updates and numerous versions of calibration files used in various experiments, there is a crucial need for robust tracking and versioning mechanisms. The decentralized nature of the system also complicates collaboration and sharing, as it hinders the ability to track changes or access different versions in a straightforward manner.

(2) **Centralized and Collaborative Access:** Maintaining and updating calibration files is a daunting task, especially as the frequency of calibration increases. Scientists, each focusing on different aspects of the quantum system, generate their own calibration files which need to be accurately tracked and updated. The need for collaboration among team members further adds to the complexity. Ensuring access

to the latest versions of these extensive files is a logistical challenge, impeding efficient collaboration.

QubiCSV provides a centralized storage solution with versioning capabilities, allowing each team member to not only track their work but also access and contribute to others' work seamlessly, as shown in Figure 3.3. As an example, we observe a scenario where multiple scientists (Scientist 1 to Scientist M) interact with the database. Each scientist has the capability to create multiple branches and access branches created by others. This flexibility in accessing and contributing to different branches fosters collaborative research and data sharing. Within each branch, scientists can maintain their unique versions of the database. They have the freedom to perform various operations like inserting new data, updating existing data, and deleting data. Moreover, they can merge two different branches, enabling the combination of data sets and collaborative developments. Branches can also be renamed or deleted as per the evolving needs of the research, ensuring the database remains organized and relevant. These data versioning capabilities are important for the management of calibration data in quantum computing research. They allow scientists to track changes, revert to previous data states if necessary, and collaborate effectively with peers. The system's design, emphasizes not just the technical capability of data versioning but also its practical application in a research environment, making it an invaluable tool for scientists working with quantum systems.

3.1.3.1 Calibration Data Schema Design

Calibration data of qubit configuration are managed in a JavaScript Object Notation (JSON) format, which can be either in a JSON file or as an individual object in a Jupyter Notebook. This quantum computing calibration dataset details the parameters governing qubits and gates within the quantum processor. Key attributes include

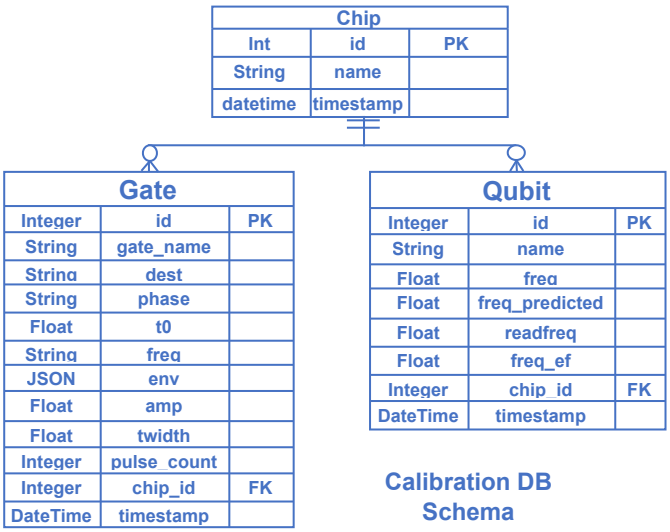

Figure 3.4. The schema for calibration data consists of three primary tables: chip, qubit, and gate. In this schema, each 'chip' is designated as a primary key (PK) and serves as a foreign key (FK) in both the qubit and gate tables. This setup allows one chip to be associated with multiple qubits and gates..

the qubit drive frequency ($freq$) and qubit readout frequency ($readfreq$). Additionally, the file outlines gate configurations. For instance, the X90 Gate for Q0, specifying frequency ($freq$), phase ($phase$), destination ($dest$), time width ($twidth$), start time ($t0$), amplitude (amp), and an envelope function (env) like "cos_edge_square" with a 25% ramp fraction for rising and falling edges. This comprehensive configuration extends to various qubits, providing detailed settings for each in terms of their drive and read frequencies, gate operations, and associated envelope functions.

The database schema of our platform is intricately designed to accommodate the specific needs of calibration data. Utilizing Dolt, a data versioning system built upon a traditional SQL database structure, we have established a schema that efficiently organizes calibration details for every chip. In this schema, the chip acts as a foreign key in both the gate and qubit tables. The detailed structure of our database schema is illustrated in Figure 3.4.

Features	Function Description	Code Example
Create Branch	*Users can create a new branch by specifying the branch name, owner's name and email, and a description. After creating a branch, they can select the appropriate chip and upload the calibration file. Physicists have the flexibility to create multiple branches and access any existing branches*	calibration.createbranch (commit_data, branch_name)
Merge Branch	*Physicists can execute merges by specifying details such as from_branch, to_branch, owner, and a message. Upon initiating the merge, all the data from the from_branch is seamlessly integrated into the to_branch, resulting in the creation of a new merge commit*	calibration.mergebranch (from_branch, to_branch, author_name)
Rename Branch	*This feature enables the modification of branch names as needed, accessible both through the UI and the Python library. This flexibility allows physicists to change the branch name to suit their current experiment or specific requirements*	calibration.renamebranch (old_branch_name, new_branch_name, author_name)
Copy Branch	*Mirroring the 'clone' function found in Git, this feature enables users to replicate an entire branch into a new branch. This capability is particularly useful for preserving the original data state while experimenting with variations in calibration.*	calibration.copybranch(branch_name)
Delete Branch	*This feature allows for the removal of unwanted data branches. This action requires specifying the branch name for confirmation, ensuring that branches are not deleted unintentionally or without proper authorization*	calibration.createbranch (branch_name, author_name)
History of Repository	*This function provides a comprehensive log of all activities at the database level. It meticulously tracks key actions such as the creation, deletion, or renaming of branches, as well as recent commits*	calibration.history( )
Commit Data	*In our system, users upload a calibration file for a chosen chip, followed by a commit operation where they enter author details and a commit message. Successful uploads generate a SHA-256 hash as a unique commit identifier.*	calibration.insertbyfile(file_path, commit_data, branch_name, chip_id)
View Calibrated Data	*After committing a calibration file, users can instantly view it on the interface and access a table of their data. In Jupyter notebooks, a JSON file named after the commit hash (e.g., Commit_Hash.json) is generated, containing the full commit data.*	calibration.getcommitdetails (commit_hash, branch_name)
Data Diff	*This tool allows physicists to compare commits within a branch, visualizing calibration data differences. By using the Data Diff function, they can closely inspect changes across commits, aiding in identifying the best calibration combination*	calibration.getcommitdiff (commit_hash, branch_name)

Figure 3.5. Table 1: Comprehensive overview of QubiCSV data versioning features for calibration data management. It includes feature descriptions and Python code examples for easy implementation in Jupyter Notebooks. This design allows seamless database interactions from notebooks, mirrored on the platform's dashboard..

3.1.3.2 Implementation of Calibration Data Management

A dashboard resembling a traditional Git interface facilitates data management, as illustrated in Figure 3.6. These screenshots provide a depiction of how users interact with the system. The interface includes a home page that serves as the entry point to the platform. From there, users can navigate to a dashboard specifically designed for managing branches, exploring individual branches or viewing details of specific commits. Additionally, the platform offers a feature to compare two commits, allowing users to easily identify differences and track changes over time. This

visual overview underscores the platform's user-friendly design and functionality in managing and visualizing calibration data in quantum computing research. The key functionalities are detailed in Table 1, which outlines the comprehensive features and capabilities of the calibration data management system within the QubiCSV platform. The table also includes code snippets for each functionality, providing users with practical examples of how to utilize these features within the Python library.

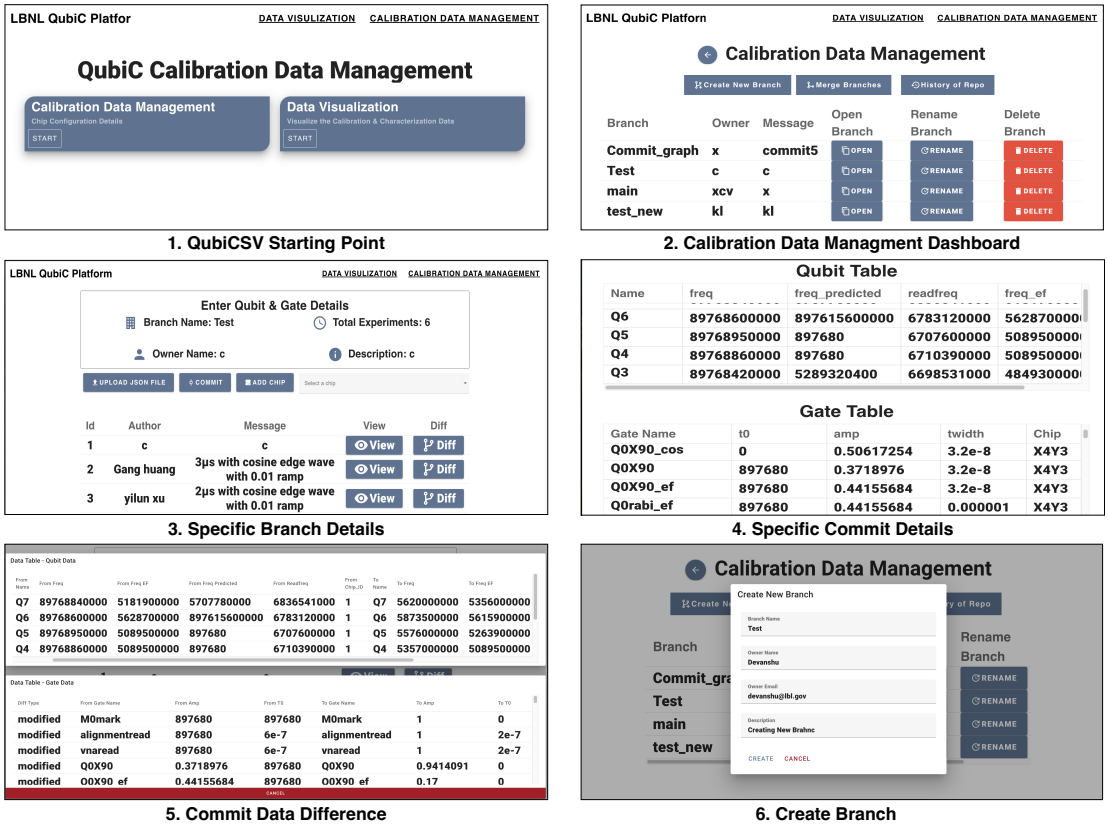

Figure 3.6. QubiCSV's dashboard screenshots. This image presents a collection of screenshots showcasing the user interface of the QubiCSV platform, providing a visual overview of the system's features and user interactions..

3.1.4 Characterization Data Management

Characterization data is a critical output generated after the calibrated QubiC system measures qubits from quantum computers. This data file contains various properties for each qubit, offering valuable insights into their performance. The key properties include ***prep0read1, prep1read0, rb1qinfidelity, separation, t1, t2ramsey, and t2spinecho***. Format of characterization data file typically follows the naming convention *Chip_name.data.json*. In our platform, the chip name is extracted directly from the filename, streamlining the process of data identification and retrieval.

1. Readout Fidelity (***prep0read1***, ***prep1read0***): The ***prep0read1*** and ***prep1read0*** metrics offer insights into the fidelity of state preparation and measurement, which are crucial for ensuring the reliability of quantum operations.

2. Randomized Benchmarking Infidelity (***rb1qinfidelity***): The ***rb1qinfidelity*** metric is a key indicator of the infidelity of single-qubit gates. It plays a crucial role in assessing the quality and precision of quantum operations, providing insights into how accurately these gates can manipulate the state of a qubit without introducing significant errors.

3. Qubit readout ***Separation***: ***Separation*** is the distance between qubit blobs, which shows how clearly different qubit states can be identified from each other.

4. Coherence Times (***t1***, ***t2ramsey***, ***t2spinecho***): The coherence time refers to the duration during which a qubit can retain its quantum state, essentially representing the lifespan of a qubit. The ***t1*** measures the time it takes for a qubit to relax to its ground state, while ***t2ramsey*** and ***t2spinecho*** gauge the qubit's dephasing time, reflecting how long it maintains its quantum state coherently.

For long-term monitoring data, specifically the characterization files crucial for understanding experiment outcomes, we opted for a NoSQL MongoDB [28] database. This decision was guided by the nature of the experimental result files, which are stored in JSON format. MongoDB, being a NoSQL database, naturally supports JSON data, making it an obvious choice for our requirements. Its flexible schema and powerful querying capabilities greatly facilitate the utilization of these JSON files for visualization purposes. Furthermore, MongoDB's scalability and performance efficiency make it an ideal fit for handling the extensive datasets typical in quantum experiments. This setup in MongoDB enables the efficient tracking of all experimental results for each qubit, thereby maintaining a comprehensive record of research progress. MongoDB can also handle high volumes and can scale both vertically or horizontally to accommodate large data loads, due to its scale-out architecture[29].

CHAPTER 4

QubiCSV Visualization

4.1 Introduction

Visualization plays an important role in enhancing the understanding and monitoring of quantum experiments. It allows researchers to observe and analyze complex quantum data in an intuitive manner. Through visualization, patterns and insights that might otherwise remain obscured in raw data can be brought to light, facilitating a deeper understanding of quantum phenomena. In the context of QubiC, effective visualization tools can transform how scientists interact with and interpret quantum data, leading to more efficient and insightful experiments. Initiatives such as VACSEN [30] and QVis [31] have made significant strides, focusing on the visualization of quantum errors and noise. These tools have laid a foundational framework for understanding quantum system behaviors, which our platform builds upon and expands. Additionally, IBM 's [32] work with the IBMQ calibration database represents another pivotal contribution, offering a comprehensive approach to managing and visualizing calibration data in quantum computing systems. These developments [33, 34] collectively inform and inspire our approach, contributing to the broader landscape of quantum computing visualization and data management. However, one key limitation of these existing tools is their specialized focus on noise and error visualization, which, while crucial, does not encompass the entire scope of data visualization needs in quantum computing research.

Visualization plays a crucial role in improving the comprehension and monitoring of quantum experiments. It allows researchers to observe and analyze complex

quantum data in an intuitive manner. Through visualization, patterns and insights that might otherwise remain obscured in raw data can be brought to light, facilitating a deeper understanding of quantum phenomena. In the context of QubiC, effective visualization tools can transform how scientists interact with and interpret quantum data, leading to more efficient and insightful experiments. From our thorough analysis, Vuetify [35] web framework is selected for its sleek design capabilities and ease of integration. Vuetify's extensive component library allowed us to create a user-friendly interface, which simplifies the complex process of analyzing and interpreting calibration data, thereby enhancing the understanding of the quantum system's performance.

4.2 Data Visualization: Motivation

For visualizing experiment results, the key requirement was a tool capable of rendering complex data plots dynamically. Plotly.js [36] emerged as the ideal choice for this purpose, due to its advanced graphical options and interactivity. It enables users to delve deeply into the experiment outcomes, facilitating a detailed and nuanced exploration of the data. To support this front-end capability, we needed a robust back-end solution. Here, Flask [37] was chosen for its simplicity and efficiency as a Python web framework. Flask's ability to handle data processing and API management made it the perfect match for our system's back-end requirements.

These decisions in our design process were driven by the need to balance functionality with user accessibility. Vuetify's user-centric design approach with automatic tree shaking, an easy-to-learn API, server-side rendering (SSR), progressive web app (PWA) support, and mobile app support, among other features. It also offers internationalization support, right-to-left (RTL) text support, and a blazing-fast framework experience. These features make Vuetify a versatile and efficient choice for building

modern, high-performance web applications and Plotly.js's [36] advanced plotting capabilities, it is a declarative charting library that offers over 40 chart types, including sophisticated options like 3D charts, statistical graphs, and SVG maps. This flexibility makes it the best choice for creating interactive, high-quality visualizations, combined with Flask's back-end proficiency [38], culminating in a broad visualization system.

The database structure we adopt is designed to efficiently organize and store characterization data for each qubit. The structure is as follows: each entry is identified by an id and is associated with a specific qubit. The ExperimentData field is an array of objects, each representing a set of characterization data linked to a particular chips. When a user uploads a new experiment.json file, the system is designed to add the new characterization data to the respective qubits and append it to the ExperimentData object array. This approach ensures that all characterization data is systematically recorded and easily retrievable for each qubit, enabling comprehensive tracking and analysis over time.

4.3 Visualization of Calibration Data

To visualize the calibration data, users first select the desired branch from the database. This step is crucial as our system incorporates data versioning, allowing for a detailed historical perspective of the data. After selecting a branch, users are presented with a list of all available chips, along with their respective properties [39]. From this point, users can choose which aspects of the calibration data they wish to explore visually. Our visualization feature offers two primary types of charts, each providing unique insights into the calibration data:

Charts By Commit (Branch and Chip Specific): This type of chart allows users to visualize all qubits and gates characteristics for individual commits within

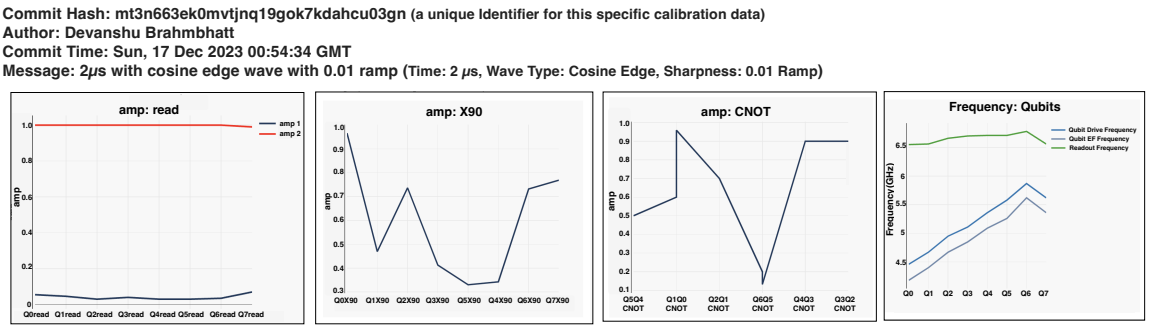

Figure 4.1. Visualization of gate groups & qubits for a specific commit in QubiCSV. Detailed readings of amplitude values for all read gates, X90 gates, and CR(cross resonance) gates are showcased for a selected commit. The chart also displays the qubit drive frequency, qubit e-f transition frequency, and readout frequency for all qubits, offering insights into their operational characteristics for the selected commit.
.

a selected branch and chip. It provides a snapshot of specific calibration states over time, enabling users to track changes and identify trends or anomalies in the calibration process. We visualize key qubit characteristics such as readout frequency, qubit drive frequency, and qubit e-f transition frequency. Moreover, we showcase essential gate characteristics including phase, frequency, amplitude, and time width. An example of these charts is shown in Figure 4.1. For instance, the graph clearly displays the amplitude values of all X90 gates for a particular commit, offering insight into the intensity of the signal used in gate operations. Similarly, the chart includes values for readout and CR gates. Additionally, the visualization extends to qubit frequency values, where the chart showcases the frequencies of all qubits for the same commit. This comprehensive display of both gate and qubit values at a specific commit point is important in understanding the calibration process's intricacies and effectiveness: Considering the wide variety of gates, we've grouped them into categories based on similar characteristics.

This methodology not only enhances the organization of the data but also significantly improves the clarity and effectiveness of our visual analysis. For example, we categorize all readout gates into the *ReadGroup*, all 90-degree rotation (along the X-axis) gates into the $X90Group$.

Charts By Properties (Commit Specific): For a given property, these charts display all the commit values for individual gates and qubits. This approach is especially valuable for examining the evolution of specific properties across different commits, providing a deeper insight into the dynamics of the calibration data. Figure 4.2 shows an example of these charts, which provides a comprehensive view of how specific properties change over time for each commit.

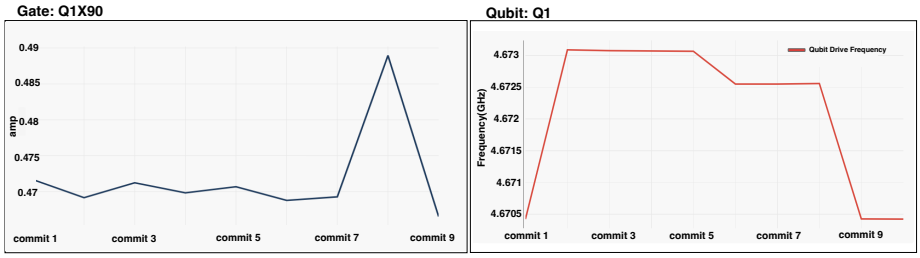

Figure 4.2. Detailed frequency and amplitude visualization in QubiCSV. This figure illustrates the X90 gate amplitude, and qubit drive frequency values for Q1 across all commits, offering a view of specific qubit and gate behavior over time..

- ***Qubits***: We have detailed property-specific visualizations for qubits. This visualization approach provides individual property charts for each qubit, showcasing how properties like qubit drive, e-f transition, and readout frequency have evolved over different commits.
- ***Gates***: Similarly, for gates, these graphs offer a comprehensive view of the evolution of gate properties such as phase, frequency, amplitude, and time width over time. Users can analyze these property-specific charts for each gate, en-

abling them to closely monitor the behavior and performance of the gates across various commits. This level of detailed visualization aids in identifying patterns, trends, and potential areas for optimization in the gate operations.

This approach demonstrates how we plot graphs for each gate, showcasing the dynamic nature of our system. It is designed to seamlessly accommodate new gates and qubits as they are added to the calibration file. With data versioning and visualization, QuiCSV provides the most effective monitoring and analysis capabilities to scientists engaged in quantum research work. This feature enhances the flexibility and adaptability of our platform, ensuring it remains a valuable tool in the ever-evolving field of quantum computing.

4.4 Visualization of Characterization Data

Characterization Data, generated after calibrated QubiC interacts with qubits, plays a pivotal role in comprehending the effectiveness of quantum experiments. It offers insights into how each qubit responds to calibration and reveals patterns in experimental behaviors. Identifying these patterns is crucial for determining the optimal calibration combinations that yield the best results. Our platform provides two distinct approaches to visualize this characterization data, catering to different analytical needs. To begin visualization, users must first select the specific chip whose characterization data they wish to analyze. This initial step ensures that the subsequent data visualizations are tailored to the selected chip (Figure 4.3).

- **By Qubits**: This method concentrates on a particular qubit, enabling users to monitor and analyze all properties associated with that qubit across various experiments. Researchers can scrutinize changes and trends in properties such as prep0read1, rb1qinfidelity, t1, t2ramsey, and others, across different experi-

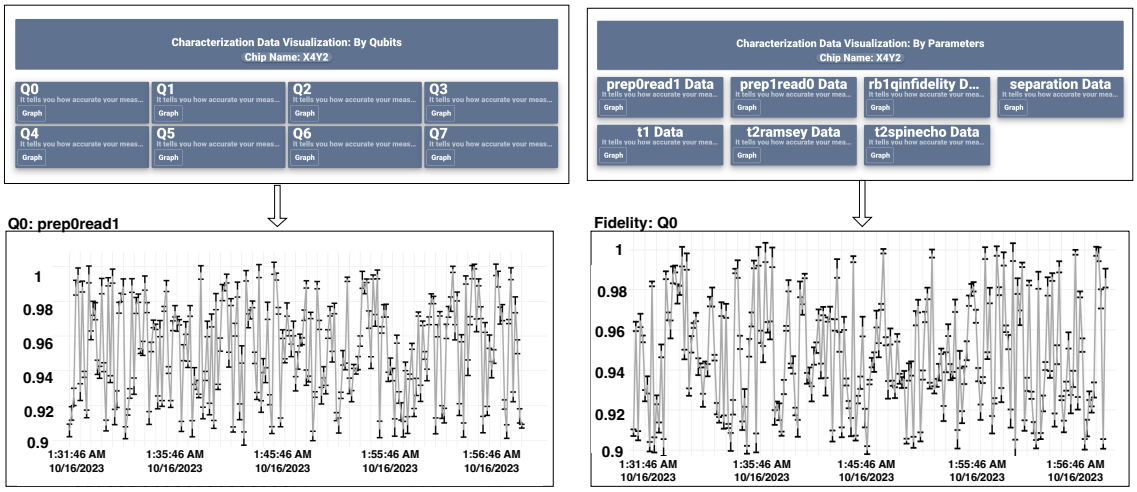

Figure 4.3. Examples of visualization of data by qubits and properties..

ments. Furthermore, researchers can pinpoint specific conditions under which the qubit operates optimally or displays anomalous behavior.

- **By Properties**: Alternatively, users can choose to focus on a specific property and observe how all qubits respond to this property across multiple experiments. This approach is particularly useful for analyzing how a specific property, like coherence time or readout errors, varies across different qubits and for identifying patterns or anomalies that are consistent or variable across the qubit array.

Both these visualization methods are designed to provide in-depth insights into the experimental data, assisting researchers in making informed decisions about future experiments and calibrations.

4.5 User Accessibility Interaction Interface

In the context of QubiCSV, 'User Accessibility Methods' refers to the various ways in which users can interact with and access the platform. The users interact

with the platform to manage and visualize quantum computing data, specifically calibration and characterization data. The frequency of interaction varies, with some users accessing the system daily for active experiments while others might use it less frequently for data review or research purposes. The design of the QubiCSV user interface was a complex task, particularly given the diverse needs and preferences within the scientific community. Initially, our primary focus was developing a Python library, considering scientists' widespread use of Jupyter Notebook for managing calibration files.

CHAPTER 5

Conclusion

We aimed to make QubiCSV an open-source platform accessible to a broader range of users including other research facilities, educational institutions, and individual researchers, we recognized the need for a more inclusive approach. A Python library, while efficient, might not cater to all potential users, especially those who are not as familiar with coding or prefer a more interactive interface. This led us to consider the advantages of a web-based platform, which could offer a more user-friendly and visually intuitive experience. A web-based interface would not only be beneficial for new or less technically inclined staff but also for the wider research community who might prefer a more graphical interface for data visualization and management.

Consequently, we opted for a dual-interface approach. We introduced a web-based platform to enhance the user experience with advanced visualization capabilities and a more intuitive interface. Simultaneously, we developed a Python library that seamlessly integrates with Jupyter Notebook. This library facilitates the same API calls for storing and retrieving data from the database, ensuring that users comfortable with Jupyter Notebook can continue to work within their preferred environment. This dual-interface approach ensures that our platform accommodates the varying needs and preferences of the scientific community. This design decision, therefore, not only caters to the immediate needs of LBNL scientists but also positions QubiCSV as a versatile tool for the broader quantum computing research community.

5.1 Contribution & Performance

Our platform offers sophisticated storage and visualization capabilities for each calibrated gate and qubit. For every data insertion, we provide detailed and interactive charts. These plots feature built-in screenshot capabilities, zoom-in-out, panning, and auto-scaling, catering to the diverse needs of data analysis.

The platform boasts an API response time of less than 500 ms. The adoption of the MVC architecture offers flexibility, particularly in modifying the user interface. This architectural choice means that if someone wishes to use our APIs to create their own visual board with modifications, they can easily do so, allowing for customization and adaptability to individual research needs. Regarding the performance of database versioning, we base our assessment on comparisons with MySQL using standard sysbench [40] metrics. Dolt, being MySQL compatible, provides a relevant benchmark for performance evaluation. Currently, Dolt's performance is approximately 1.9X slower than MySQL: 1.3X slower for write operations and 2.3X slower for read operations, as per the standard suite of sysbench tests. Despite being slower than MySQL, with most MySQL queries returning in the 0-10ms range, Dolt's performance remains within a practical range for user applications, especially considering its versioning capabilities.

This Chrome Developer Tools Performance tab screenshot captures a detailed performance profile of a web platform as shown in Figure 5.1, especially under a 4x CPU slowdown simulation, which is often done to mimic less powerful devices and ensure that the web application performs well even on lower-end hardware. The performance analysis of our web application over a 30-second period reveals a timeline of color-coded activities, each representing different browser tasks—scripting in yellow, rendering in purple, painting in green, and system operations in gray—with red triangles highlighting long tasks that may impede responsiveness. A detailed flame

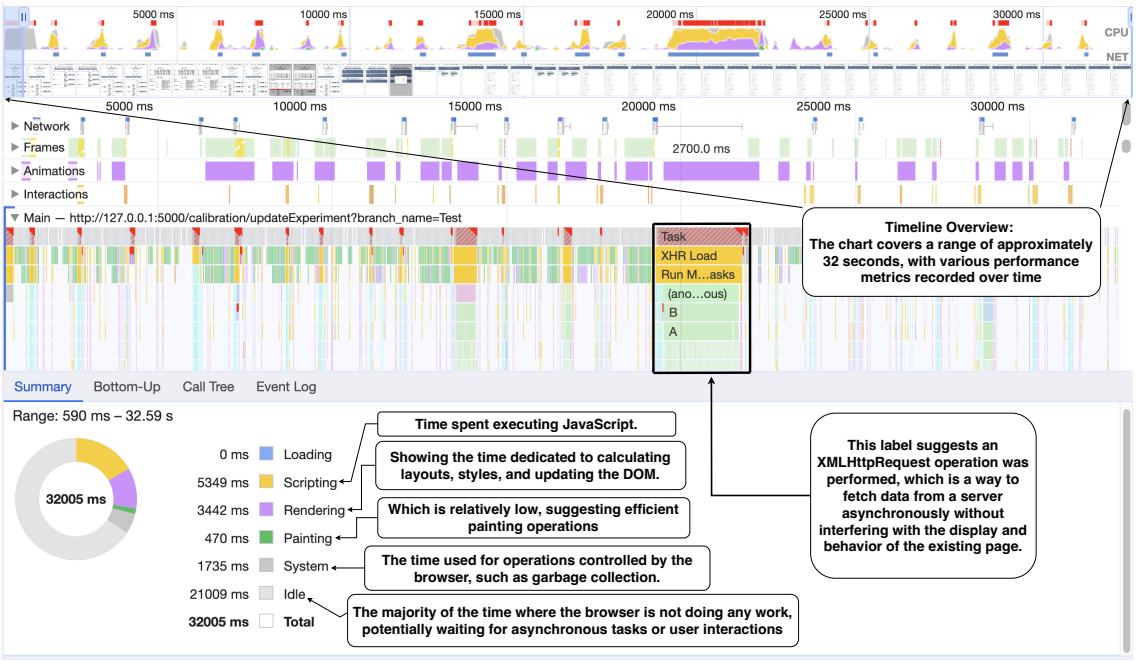

Figure 5.1. Over a 30-second period under a 4x CPU slowdown, the platform demonstrates swift handling of complex visualizations with Plotly, maintaining fast scripting (5129 ms) and rendering (3375 ms) amidst substantial idle time, indicating a responsive system optimized for heavy data operations, it demonstrates the web platform's efficiency in managing complex visualizations and API data fetching, maintaining responsiveness and good performance metrics obtained from Google Chrome..

chart illustrates the JavaScript call stack, pinpointing performance bottlenecks, while network requests indicate the timing of data transmission crucial for API performance assessment. The frame rate and rendering times show the efficiency of style recalculations and screen drawings. The summary pie chart at the bottom details the time allocation across operations, with the majority being idle time (21,090 ms), suggesting the system awaits completion of tasks or user input. Notably, the scripting (5,349 ms) and rendering (3,442 ms) times remain low compared to the idle time, which indicates that the web application is efficiently handling the heavy visualization and chart plotting using Plotly, even with the increased load from a slower CPU.

REFERENCES

[1] F. Jazaeri, A. Beckers, A. Tajalli, and J.-M. Sallese, "A Review on Quantum Computing: From Qubits to Front-end Electronics and Cryogenic MOSFET Physics," in *2019 MIXDES - 26th International Conference "Mixed Design of Integrated Circuits and Systems"*, June 2019, pp. 15–25.

[2] N. S. Yanofsky, "An Introduction to Quantum Computing," in *Proof, Computation and Agency: Logic at the Crossroads*, ser. Synthese Library, J. van Benthem, A. Gupta, and R. Parikh, Eds. Dordrecht: Springer Netherlands, 2011, pp. 145–180. [Online]. Available: https://doi.org/10.1007/978-94-007-0080-$2_1$0

[3] A. Matos, "Overview of Prominent Decoherence Mechanisms in Superconducting Qubits."

[4] M. Kjaergaard, M. E. Schwartz, J. Braumüller, P. Krantz, J. I.-J. Wang, S. Gustavsson, and W. D. Oliver, "Superconducting Qubits: Current State of Play," *Annual Review of Condensed Matter Physics*, vol. 11, no. 1, pp. 369–395, 2020, _eprint: https://doi.org/10.1146/annurev-conmatphys-031119-050605. [Online]. Available: https://doi.org/10.1146/annurev-conmatphys-031119-050605

[5] R. LaPierre, "Superconducting Qubits," in *Introduction to Quantum Computing*, ser. The Materials Research Society Series, R. LaPierre, Ed. Cham: Springer International Publishing, 2021, pp. 285–322. [Online]. Available: https://doi.org/10.1007/978-3-030-69318-$3_2$2

[6] M. H. Devoret, A. Wallraff, and J. M. Martinis, "Superconducting Qubits: A Short Review," Nov. 2004, arXiv:cond-mat/0411174. [Online]. Available: http://arxiv.org/abs/cond-mat/0411174

[7] H. Paik, D. I. Schuster, L. S. Bishop, G. Kirchmair, G. Catelani, A. P. Sears, B. R. Johnson, M. J. Reagor, L. Frunzio, L. I. Glazman, S. M. Girvin, M. H. Devoret, and R. J. Schoelkopf, "Observation of High Coherence in Josephson Junction Qubits Measured in a Three-Dimensional Circuit QED Architecture," *Physical Review Letters*, vol. 107, no. 24, p. 240501, Dec. 2011, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.107.240501

[8] P. Zhao, P. Xu, D. Lan, J. Chu, X. Tan, H. Yu, and Y. Yu, "High-Contrast ZZ Interaction Using Superconducting Qubits with Opposite-Sign Anharmonicity," *Physical Review Letters*, vol. 125, no. 20, p. 200503, Nov. 2020, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.125.200503

[9] Y. Yang, Z. Shen, X. Zhu, Z. Wang, G. Zhang, J. Zhou, X. Jiang, C. Deng, and S. Liu, "FPGA-based electronic system for the control and readout of superconducting quantum processors," *Review of Scientific Instruments*, vol. 93, no. 7, p. 074701, July 2022, arXiv:2110.07965 [physics, physics:quant-ph]. [Online]. Available: http://arxiv.org/abs/2110.07965

[10] D. Dong and I. R. Petersen, "Introduction to Quantum Mechanics and Quantum Control," in *Learning and Robust Control in Quantum Technology*, ser. Communications and Control Engineering, D. Dong and I. R. Petersen, Eds. Cham: Springer International Publishing, 2023, pp. 7–33. [Online]. Available: https://doi.org/10.1007/978-3-031-20245-2_2

[11] D. J. Reilly, "Challenges in Scaling-up the Control Interface of a Quantum Computer," in *2019 IEEE International Electron Devices Meeting (IEDM)*, Dec. 2019, pp. 31.7.1–31.7.6, iSSN: 2156-017X.

[12] I. Siddiqi, "Engineering high-coherence superconducting qubits," *Nature Reviews Materials*, vol. 6, no. 10, pp. 875–891, Oct. 2021, number: 10 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41578-021-00370-4

[13] A. D. Corcoles, A. Kandala, A. Javadi-Abhari, D. T. McClure, A. W. Cross, K. Temme, P. D. Nation, M. Steffen, and J. M. Gambetta, "Challenges and Opportunities of Near-Term Quantum Computing Systems," *Proceedings of the IEEE*, vol. 108, no. 8, pp. 1338–1352, Aug. 2020, arXiv:1910.02894 [quant-ph]. [Online]. Available: http://arxiv.org/abs/1910.02894

[14] "QCCS System Control | Zurich Instruments," Nov. 2020. [Online]. Available: https://www.zhinst.com/en/quantum-computing-systems/system-control

[15] "Qblox | superconducting." [Online]. Available: https://www.qblox.com/superconducting

[16] "Quantum Control System (QCS) - The world's first fully digital quantum control solution."

[17] Q. M. Team, "Quantum Control: the Key to Achieving Qubit Scalability," Dec. 2022. [Online]. Available: https://www.quantum-machines.co/blog/quantum-control-the-key-to-achieving-qubit-scalability/

[18] Y. Xu, G. Huang, J. Balewski, R. Naik, A. Morvan, B. Mitchell, K. Nowrouzi, D. I. Santiago, and I. Siddiqi, "QubiC: An open source FPGA-based control and measurement system for superconducting quantum information processors," *IEEE Transactions on Quantum Engineering*, vol. 2, pp. 1–11, 2021, arXiv:2101.00071 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2101.00071

[19] Y. Xu, G. Huang, N. Fruitwala, A. Rajagopala, R. K. Naik, K. Nowrouzi, D. I. Santiago, and I. Siddiqi, "QubiC 2.0: An Extensible Open-Source Qubit Control System Capable of Mid-Circuit Measurement and Feed-Forward," Sept. 2023, arXiv:2309.10333 [physics, physics:quant-ph]. [Online]. Available: http://arxiv.org/abs/2309.10333

[20] T. Miyoshi, K. Koike, S. Morisaka, H. Shiomi, K. Ogawa, Y. Tabuchi, and M. Negoro, "FPL Demo: A Flexible and Scalable Quantum-Classical Interface based on FPGAs," in *2022 32nd International Conference on Field-Programmable Logic and Applications (FPL)*, Aug. 2022, pp. 473–473, iSSN: 1946-1488.

[21] A. Wallraff, D. I. Schuster, A. Blais, L. Frunzio, J. Majer, M. H. Devoret, S. M. Girvin, and R. J. Schoelkopf, "Approaching Unit Visibility for Control of a Superconducting Qubit with Dispersive Readout," *Physical Review Letters*, vol. 95, no. 6, p. 060501, Aug. 2005, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.95.060501

[22] H. E. Brandt, "Qubit devices and the issue of quantum decoherence," *Progress in Quantum Electronics*, vol. 22, no. 5, pp. 257–370, Sept. 1999. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0079672799000038

[23] "Phys. Rev. Lett. 127, 180501 (2021) - Strong Quantum Computational Advantage Using a Superconducting Quantum Processor." [Online]. Available: https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.127.180501

[24] "Model-View-Controller Pattern," in *Learn Objective-C for Java Developers*, J. Bucanek, Ed. Berkeley, CA: Apress, 2009, pp. 353–402. [Online]. Available: https://doi.org/10.1007/978-1-4302-2370-$2_2$0

[25] A. Sengupta, S. Sengupta, and S. Bhattacharya, "A Framework for Component Design using MVC Design Pattern," *INFOCOMP Journal of Computer Science*,

vol. 7, no. 4, pp. 60–69, Dec. 2008, number: 4. [Online]. Available: https://infocomp.dcc.ufla.br/index.php/infocomp/article/view/239

[26] T. Dey, "A comparative analysis on modeling and implementing with mvc architecture," *International Journal of Computer Applications*, vol. 1, pp. 44–49, 2011.

[27] V. Surwase, "Rest api modeling languages-a developer's perspective," *Int. J. Sci. Technol. Eng*, vol. 2, no. 10, pp. 634–637, 2016.

[28] "MongoDB: The Developer Data Platform | MongoDB," https://www.mongodb.com/, accessed: 2024-02-04. [Online]. Available: https://www.mongodb.com/

[29] "Database Scaling," https://www.mongodb.com/basics/scaling, accessed: 2024-02-04. [Online]. Available: https://www.mongodb.com/basics/scaling

[30] S. Ruan, Y. Wang, W. Jiang, Y. Mao, and Q. Guan, "VACSEN: A Visualization Approach for Noise Awareness in Quantum Computing," July 2022, arXiv:2207.14135 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2207.14135

[31] C. Steed, J. Chae, S. Dasgupta, and T. Humble, "QVis: A Visual Analytics Tool for Exploring Noise and Errors in Quantum Computing Systems," Oak Ridge National Laboratory (ORNL), Oak Ridge, TN (United States), Tech. Rep., Sept. 2023. [Online]. Available: https://www.osti.gov/biblio/2001390

[32] "Administration," https://quantum.ibm.com/admin/hubs, accessed: 2024-02-04. [Online]. Available: https://quantum.ibm.com/admin/hubs

[33] M. Miller and D. Miller, "GraphStateVis: Interactive Visual Analysis of Qubit Graph States and their Stabilizer Groups," in *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*, Oct. 2021, pp. 378–384. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9605348/figuresfigures

[34] J. Bley, E. Rexigel, A. Arias, N. Longen, L. Krupp, M. Kiefer-Emmanouilidis, P. Lukowicz, A. Donhauser, S. Küchemann, J. Kuhn, and A. Widera, *Visualizing Entanglement in multi-Qubit Systems*, May 2023.

[35] "Vuetify — A Vue Component Framework," https://vuetifyjs.com/en/, accessed: 2024-02-04. [Online]. Available: https://vuetifyjs.com/en/

[36] "Plotly javascript graphing library in JavaScript," https://plotly.com/javascript/, accessed: 2024-02-04. [Online]. Available: https://plotly.com/javascript/

[37] "Welcome to Flask — Flask Documentation (3.0.x)," https://flask.palletsprojects.com/en/3.0.x/, accessed: 2024-02-04. [Online]. Available: https://flask.palletsprojects.com/en/3.0.x/

[38] "The Ultimate Guide to Improving Flask Performance | Scout APM Blog," https://scoutapm.com/blog/improving-flask-performance, accessed: 2024-02-04. [Online]. Available: https://scoutapm.com/blog/improving-flask-performance

[39] R. Chatterjee, G. Arun, S. Agarwal, B. Speckhard, and R. Vasudevan, "Using data versioning in database application development," in *Proceedings. 26th International Conference on Software Engineering*, May 2004, pp. 315–325, iSSN: 0270-5257. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1317454?casa$_token$ =$_7$ *kqbLZ* − *JRIAAAAA* : *zUgOSDx*0*OSMFp*2*ZSXF*8*vZMCXDot*9*BoHRC*0*DetoHqbDl*8*cMXLWwQADk*6*j*

[40] "Sysbench," https://en.wikipedia.org/wiki/Sysbench, accessed: 2024-02-26.

BIOGRAPHICAL STATEMENT

Devanshu Brahmbhatt, born in India, is an aspiring entrepreneur and a Graduate Research Assistant at the WSSL Lab, pursuing his MS in Computer Science at The University of Texas at Arlington, due to graduate in May 2024. He has interned at Lawrence Berkeley National Lab, focusing on quantum computing, and is currently the LLM Product Manager at Skyflow, California, enhancing data security through innovative guardrails. Devanshu co-founded PaperTalk.io, a platform that increases the accessibility and engagement of academic research, growing to over 4,000 users. His professional skills blend product management with technical expertise in machine learning and blockchain technologies. Devanshu is passionate about blogging and technological innovation, regularly contributing to GeeksforGeek and Medium on topics like product development and AI.