Summary of work for IP on Python Ecosystem
MONSOON SEMESTER 2022

INSTRUCTOR: Dr. Pankaj Jalote

GROUP TOPIC: Quantum Computing using Python

GROUP MEMBERS: Bhagesh Gaur (2020558) & Ojus Singhal (2020094)

Credits taken for the IP: 4

Github Link

20th December, 2022

Index

1. Overview

- **Quantum Mechanics:**
  - Completed this course: Introduction to Quantum Computing with Qiskit (with IBM Quantum Research Zurich)
  - Watched and read algorithmic tutorials such as these.
  - Looked at the open-source libraries for Quantum Computing. Finally, went ahead with Qiskit SDK and wrote programs to learn the structure of Qiskit code.
  - Started trying to implement quantum circuits and algorithms on our own, along with circuit visualization. Code
  - Read several Quantum Computing research papers, blog posts and articles.
  - Completed our major project in quantum computing
  - Completed other assignments

- **Other Groups**
  - Attended most of the presentations and tried to implement the relevant code for each presentation ourselves.
  - We utilised these concepts in some of course projects and assignments throughout the semester. The discussions on ASTs and Functional programming were two of the most intriguing to us.

2. Assignments Completed

- **Parallel and Distributed processing with Python**
  - We completed the assignment provided for the Parallel and Distributed processing topic.
  - We learnt about use of the concurrent.futures and ThreadPoolExecutor in the first question and about executing parallel threads using the submit function. In this question we observed a significant difference in overall runtime from nearly 13 seconds to almost 2 seconds using parallel programming.
  - In the second question we utilized the mpi4py library, using it we got the core counts, split a large array of random numbers and then used different cores for each list sorting operation.
  - In the third question we learnt about the distributed processing framework, Hadoop. We got to know the notion of using the mapper and reducer in operations for parallelising the task at hand.
  - This assignment gave some great insights into how we can use parallel processing in our python projects.
- **Quantum computing with Python**
  - While we made this assignment, we learnt how to structure assignments in such a manner that using some fundamental concepts how can one solve questions that require their extrapolation.
  - We solved this assignment by raising the difficulty of the task at hand, especially in the last question where we made a custom quantum gate, and ran it on actual quantum computer.
- **Frameworks and Packages**
  - We partially completed this assignment, but due to logistical constraints and lack of support from Wheels, and other packages for M1 macs we were unable to complete this assignment.
  - We learnt about framework management and creation in this assignment. We also gained insights into how to publish our own frameworks.
  - This assignment introduced us to how to get started with making our own frameworks, since there is a lot of scope in making frameworks for different applications and even workflows.

3. Project Work

Quantum Prototype-Based Clustering

(Quantum K-means Clustering)

- ***Introduction***

  While working on alot of machine learning projects and assignments in the current and past semesters, one thing that we noticed was that many of the machine learning algorithms are quite slow, and most of the time it is due to the large datasets or due to the iterative nature of the process to run over and over again till the problem converges. Quantum Computing offers better solutions for optimization problems, this property can be exploited in ML. We took a common clustering algorithm, k-means clustering and implemented it's quantum counter part. This helped us compare the two approaches and also analyse whether it is feasible to utilise the resources from quantum computers for it. We implemented the Quantum K-means concept and algorithm provided by this research paper using Qiskit SDK by IBM Quantum.

- ***Dataset***

  We used the common Iris Dataset(only 2 features considered) with 3 classes and a randomly generated dataset with 2 feature coordinates and 3 classes.

- ***Methodology***

  In this quantum k-means algorithm we use quantum computers/simulators for distance approximations between the points and centroids. In general, k-means algorithm is quite slow to converge to large number of distance calculations, especially with larger number of points and k values. K-means algorithm's time complexity is nearly equal to O(kdn) per iteration, where k is the number of cluster, d is the number of features and n is the number of points. Instead of performing costly distance calculations using the classical way we can instead use quantum computing to do this.

- ***Result and Analysis***

  After constructing the quantum circuit according the algorithm in the paper and running both classical and quantum k-means algorithm we observed that quantum k-means performs better in many scenarios like data with noise, random data and is faster converge on large datasets. In case of randomly generated dataset quantum k-means performed

better. While in case of Iris dataset, it performed comparatively the similar to classical k-means.

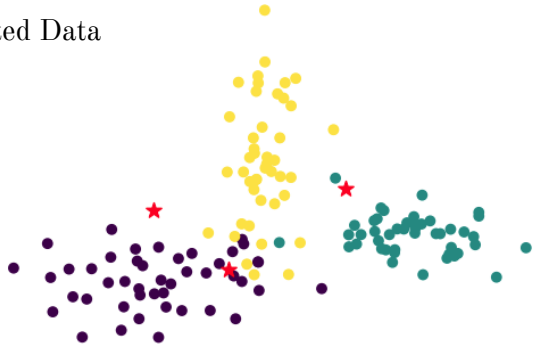
- ***Conclusions***

  While the quantum k-means algorithm has a lot of advantages like better time complexity, scalability with increased data, etc. there are a few of the drawbacks as well like sensitivity to anomalies and outliers. The hardware requirements by the quantum algorithms also pose challenges to their usage in such tasks. The Quantum K-means algorithm has a lower per iteration time complexity as compared to classical k-means.

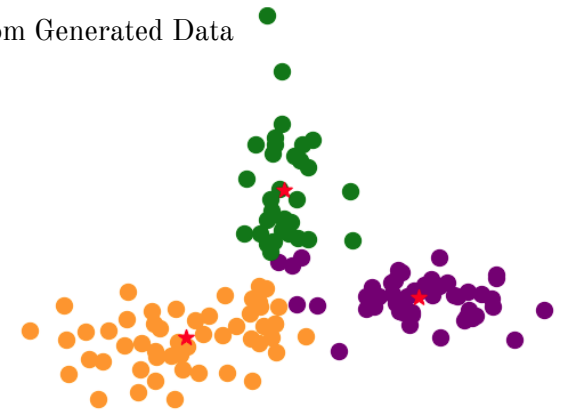
- ***References***

1. https://proceedings.neurips.cc/paper/2019/file/16026d60ff9b54410b3435b403afd226-Paper.pdf
2. https://quantumalgorithms.org/chap-q-means.html

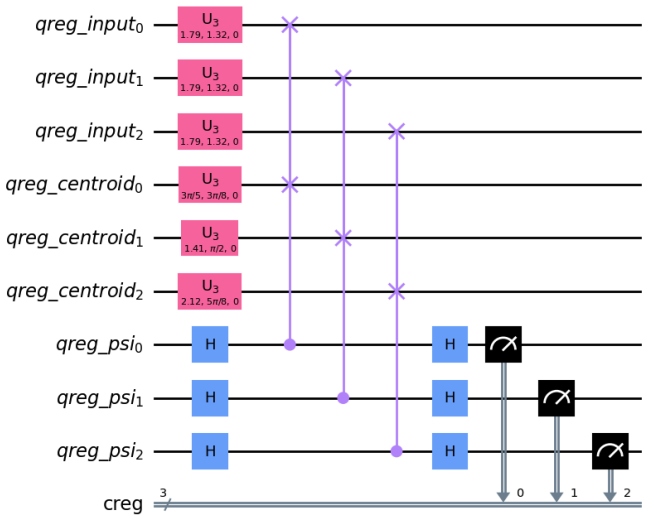
Results with Classical K-means on Random Generated Data



Results with Quantum K-means on Random Generated Data



Quantum Circuit used we need 3 quantum registers, of size k one for a data point (input), one for each centroid.



Exploring Quantum Hardware

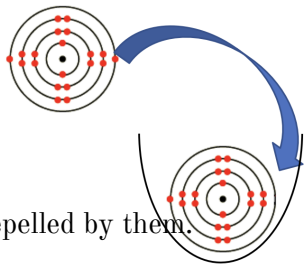
(Trap-Ion Qubits)

- ***Introduction***

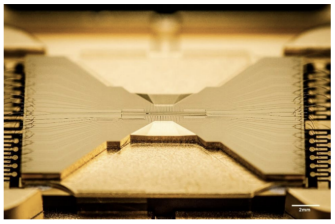
  While exploring quantum computing and various quantum computing algorithms during the semester, we became fascinated with the sophistication and beauty of quantum hardware. So we decided to explore various methods used to develop quantum hardware and their underlying fundamentals, and finally explored one of them(Trap-Ion) in detail.

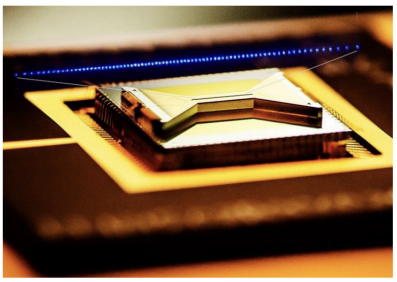
- ***Our Findings***
  - The fundamental requirements for running real-world quantum algorithms are:
    - having many qubits
    - Either with no noise (error), or error-corrected
  - We cannot run grover's on a meaningful dataset today because of not having enough qubits, noisy qubits and imperfect gates.
  - DiVincenzo criteria: Set of five basic conditions for the physical realization of a quantum computer.
    - A scalable physical system with well-characterized qubits.
    - The ability to initialize the state of the qubits.
    - Long relevant decoherence times, much longer than the gate operation time. Here, decoherence times mean the amount of time that the qubits preserve their quantum properties.
    - A "universal" set of quantum gates.
    - A qubit-specific measurement capability.
  - Qubits can be improved by providing more isolation from the environment(using better materials and reduced loss mechanisms) and making transition from 1 to 0 more difficult.
  - Qubit decoherence: Quantum systems lose their coherence as a result of interacting with the environment(i.e. Quantum superposition turns into a classical probability distribution and lose their ability of quantum interference).
  - Trapped ion qubits: The most ideal and clean quantum system. Based on well understood Physics.

- Trapping charged atoms is relatively easy.
    - Step 1: Find an atom with two outer electrons
    - Step 2: Ionize it
    - Step 3: Confine it in an electromagnetic trap
- Here, these traps are electric fields since positively charged particles are repelled by them.



Trap-Ion Chips



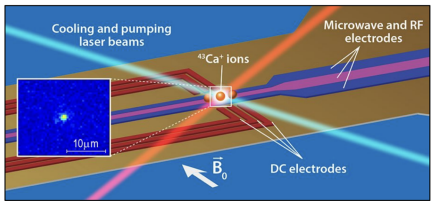


- There are pros and cons to using different qubit generator methods. Pros for using Trap ions are:
    - They have excellent quantum properties
    - Traps are made with known techniques
    - Laser and photonics technology is mature enough
    - High quality gates

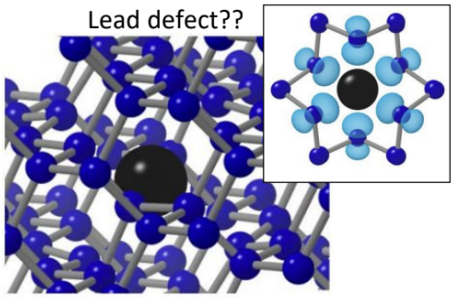
Cons for using Trap ions are:

- They have slow operation.
- They need vacuum.

- Ongoing research and developments in Trap-Ion Qubits include scaling to more qubits, achieving error correction and fault tolerance.
- Some other quantum hardware implementations are Solid-state defects centers, Photonic qubits and Superconducting qubits.



Trapped Ion



Solid-state defects centers

4. Our Learnings

4.1. Quantum Computing | We learnt about:

- The fundamentals of quantum computing from a mathematical perspective
- State-of-the-art in quantum computing and its hardware
- Python support provided by different companies such as IBM, Google, etc
- Built a fundamental understanding of Qiskit, the most popular quantum computing library. (In the process, we learned about how huge open-source libraries can be structured and maintained.)
- Learned various algorithms in Quantum computing including Shore's algorithm, Deutsch-Jozsa Algorithm and Quantum Teleportation algorithm.
- Implemented these algorithms in Qiskit.
- Learnt how to develop quantum circuits/programs for a problem from scratch.

4.2. Other Topics

- Gained an all-round deeper understanding of the python language, how it works in the backend, how libraries and packages are maintained and distributed, etc.
- Learned important lessons about how to structure, improve, maintain, distribute, and manage projects in python, along with best practices in programming and writing cleaner code.
- Learned about python runtime and cpython, along with ASTs. Gained insights on how to use functionqal programming and it's use in our python code.
- Learnt about proof of work and proof of state in blockchains and their differences.