TCS Quantum Challenge

Final Defense

Challenge – *Replenishment of retail stores*

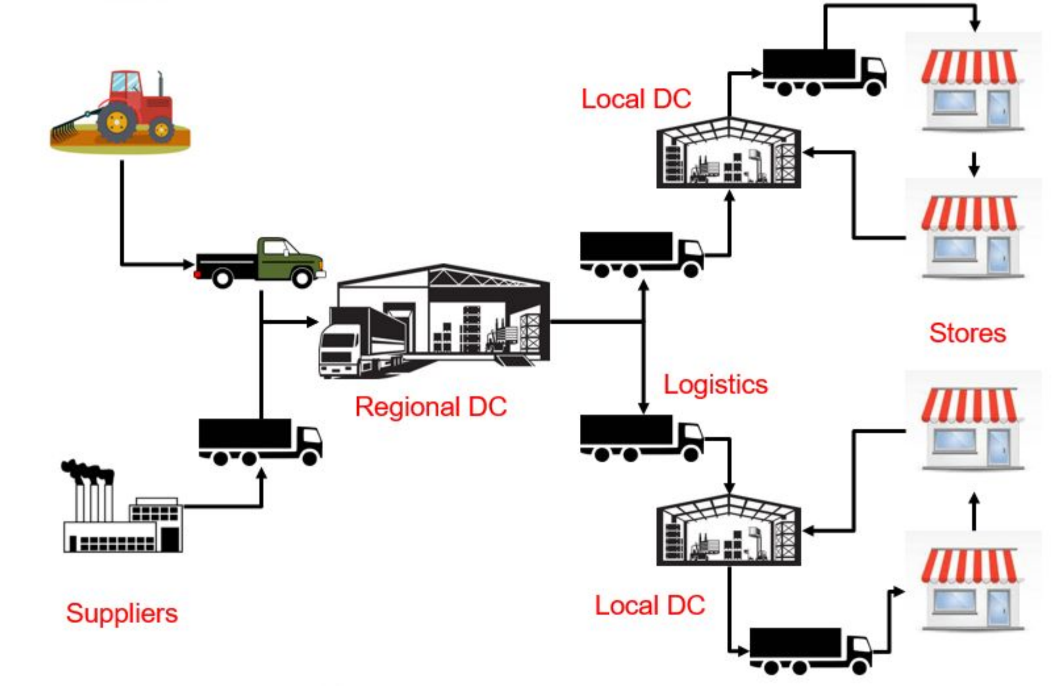
Team Lead Name : *Pratik De*
Team Members : *Grishma, Nikhil, Jay and Sreekuttan LS*
Team Entry Number : *Entry 01535209*

Problem Statement

Replenishment problem

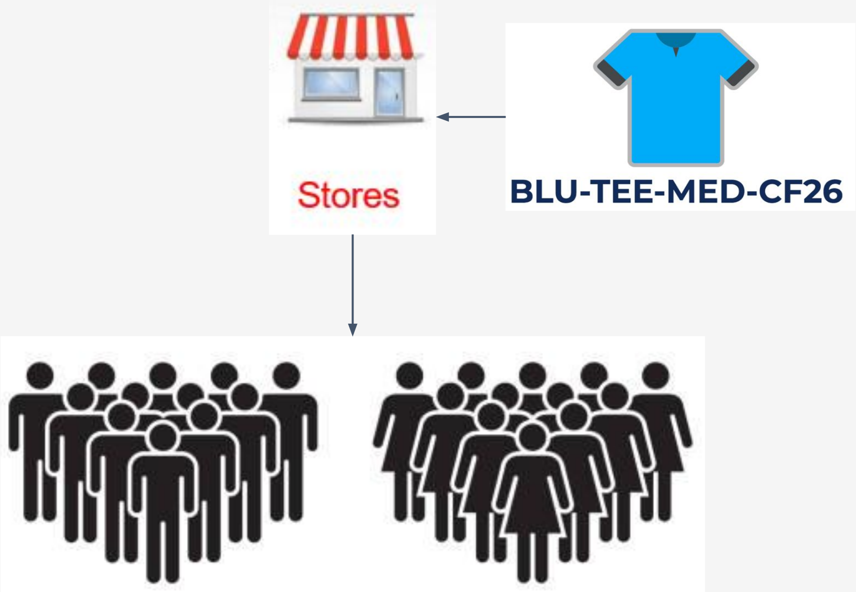
- Challenge of determining when and how much inventory to reorder in order to maintain optimal stock levels while minimizing costs.

- The Replenishment problem aims to balance inventory holding costs (e.g., storage, insurance) against stockout costs (e.g., lost sales, customer dissatisfaction).



Scope

- Level 1 Problem - Inventory management at the store level.
- **Input variables**:
  - Forecasted Demand
  - Capital
  - On Hand Quantity
- **Sale -** Probabilistic Model
- **Decision Variable**
  - ***Replenishment Quantity***
- **Objective** - Maximize Profit per cycle



Solution Overview

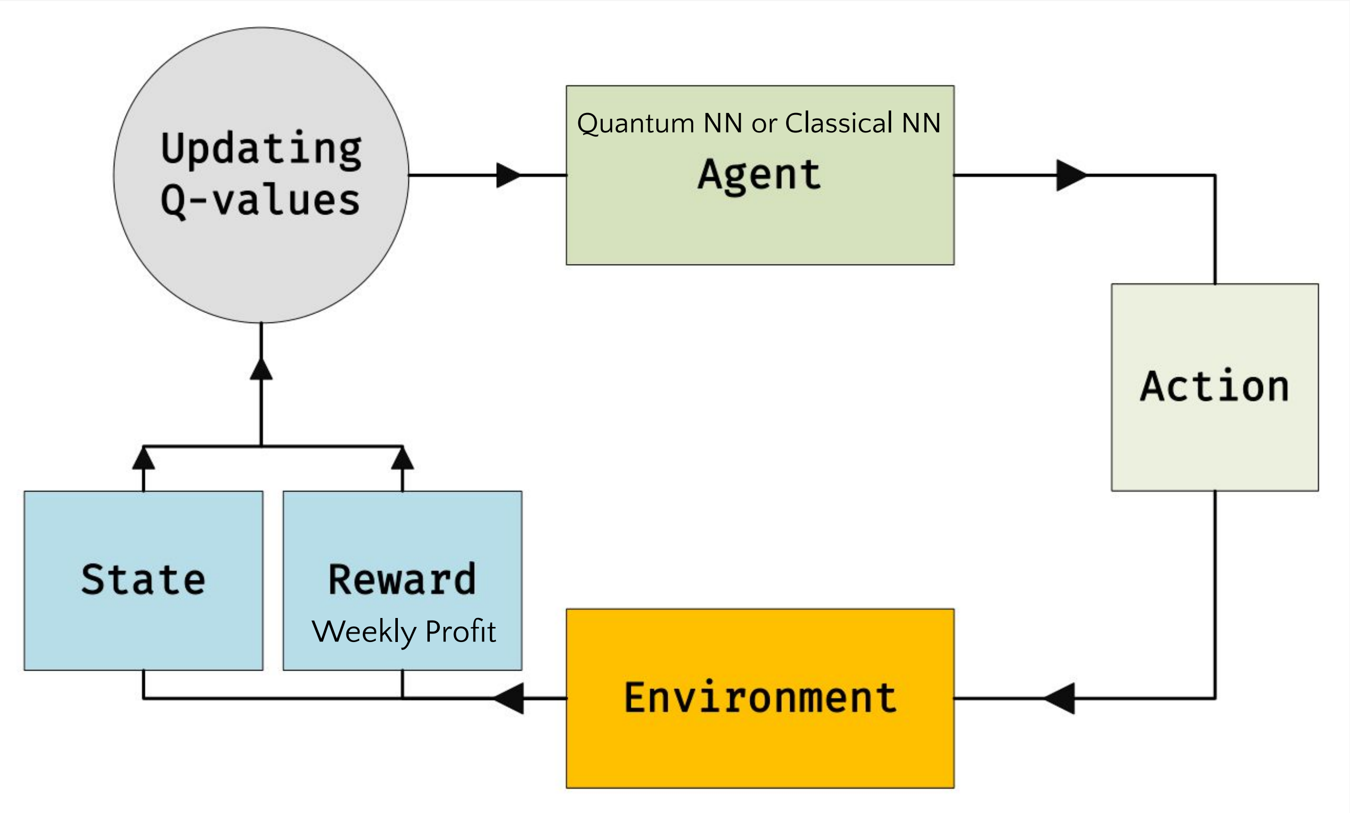
- Breaking down the problem into several problems with the objective of maximizing profit.

Simultaneously tackling several replenishment decisions. → Prioritizing product/store combinations. → Choosing single SKU and an store.

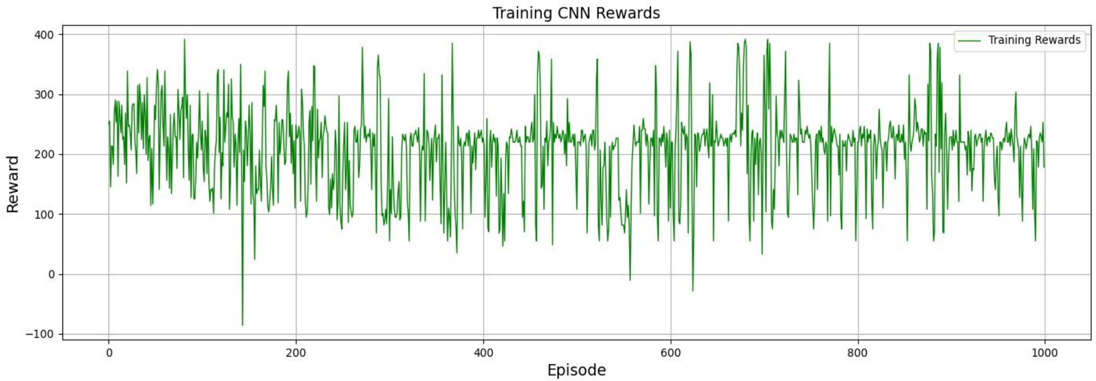
We consider a **single** **Item/SKU** and a **single** **Store**

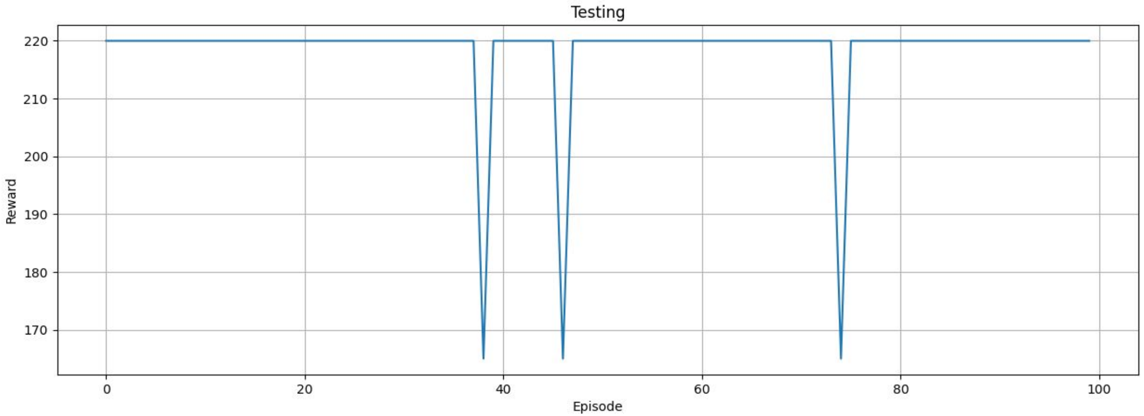
Reinforcement Learning: Q-Learning

State = (On Hand, Capital, Forecasted Demand)



Results – Classical RL

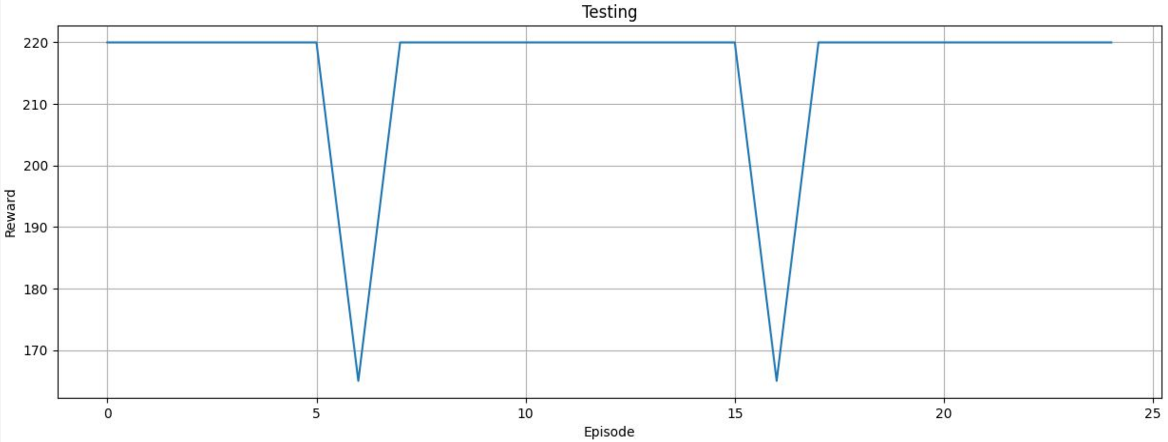


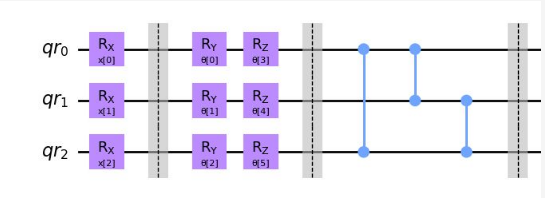



```
Model: "dqn"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               multiple                  96

 dense_1 (Dense)             multiple                  600

 dense_2 (Dense)             multiple                  100

=================================================================
Total params: 796 (3.11 KB)
Trainable params: 796 (3.11 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

- With the Adam Optimizer and Mean Squared Error
- Epsilon Greedy Policy
- **796** parameters/weights
- 1 *Episode* = 25 weeks
- 1 SKU and 1 Store
- Reward = Profit

Results - Quantum RL

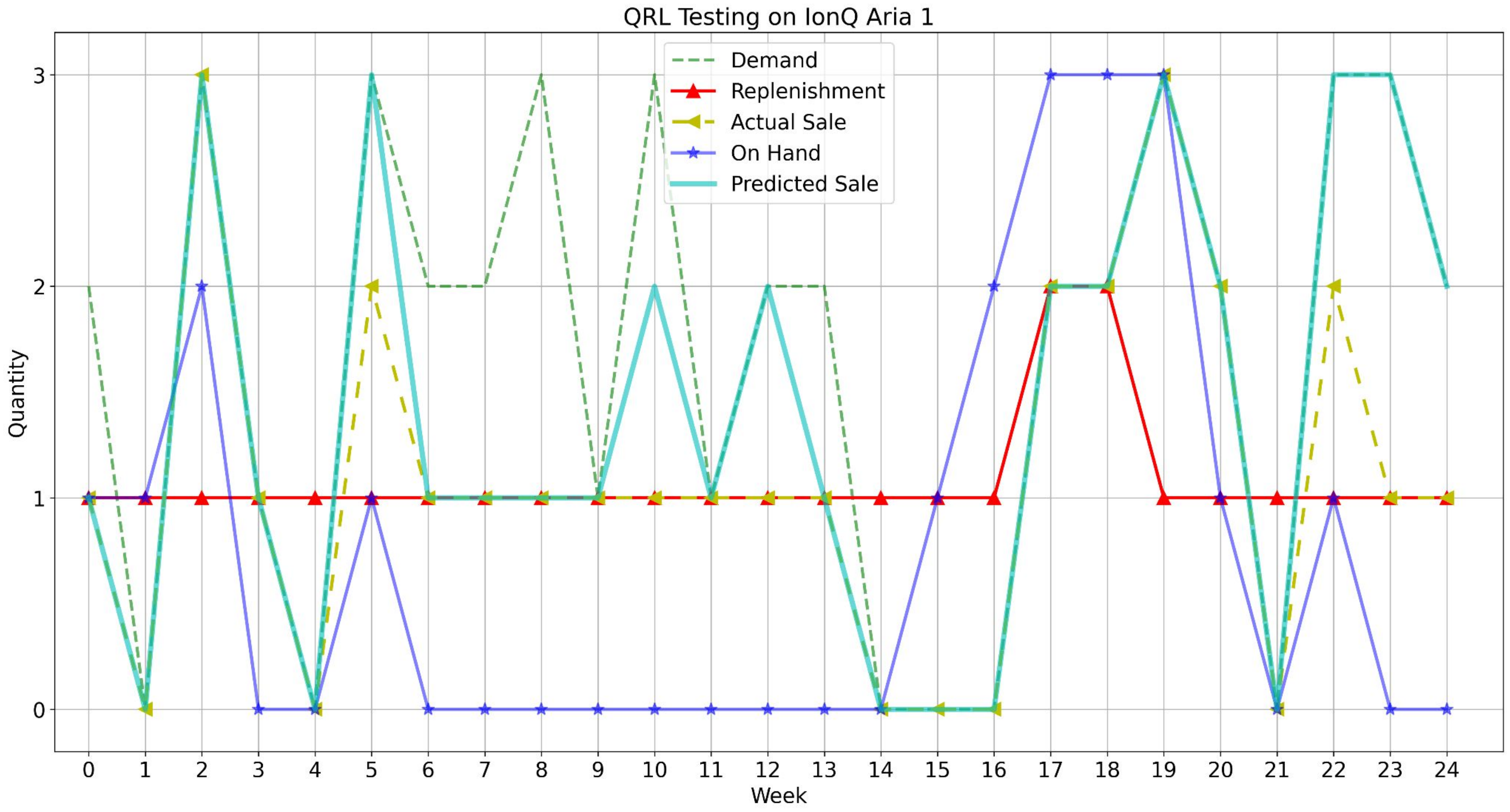




1 layer of PQC

- Parametric Quantum Circuit (PQC)
- Adam Optimizer and Mean Squared Error
- Epsilon Greedy Policy
- ***6*** parameters/weights
- 1 *Episode* = 25 weeks
- 1 SKU and 1 Store
- Reward = Profit

Results – QPU


QRL Testing on IonQ Aria 1
Demand
Replenishment
Actual Sale
On Hand
Predicted Sale
Quantity
Week
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24

Results – Comparison

Model	Best Training Reward	Testing Reward
QRL (AerSimulator)	327	218 (Mean - 25 episodes)
QRL (IonQ Aria 1)	-	206.8
CRL	391	219 (Mean - 100 episodes)

Table 1: Rewards for the graphs above.

Model	Time taken	Episodes
QRL (AerSimulator)	$\approx$ 35 mins	1000
QRL (IonQ Aria 1)	$\approx$ 3 hrs	1
CRL	$\approx$ 4 mins	1000

Table 2: Timing Models.

Business Impact

Our solution utilizes 132x less resources (parameters) than the classical counterparts!

Business Impact

QRL **on par** with Classical with an experimental dataset.

Easier to **Scale** with increasing complexity of the independent variables.

In future when QPU usage costs go down, QRL can be a profitable option.

Scalability

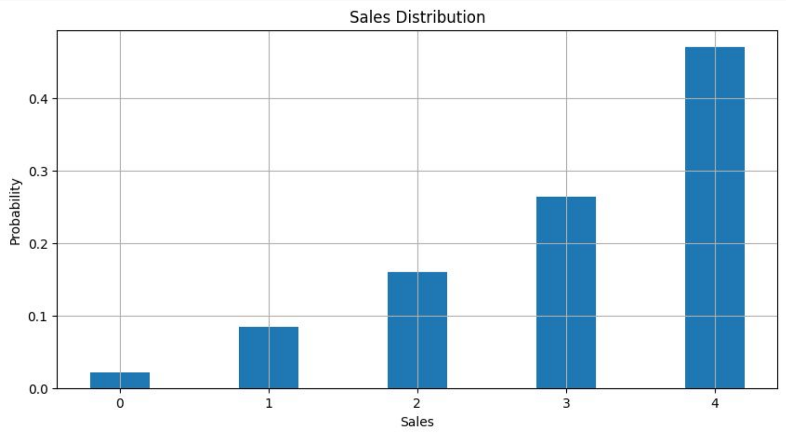
- For each new independent variable introduced, an additional qubit is required.
- Additional possible independent variables include:
  - Changing demand patterns for items through the year.
  - Similarity between different items.
  - Customer distribution across stores.
  - Distance between stores.
  - Transportation facilities to stores.
  - Distribution of customers in different categories (types of customers).
- Quantum resources will depend upon:
  - Number of product and store combinations (Iterations Scale Exponentially).
  - Number of shots for a quantum circuit.
  - For instance, **1000 actions** (replenishment choices) require **10 qubits** O(log(Actions)).

Future Roadmap

- Experimentation with more product and store combinations.
- Impact analysis with a longer forecasting horizon (**>25 weeks**).
- Performance analysis with additional data:
    - Purchasing pattern differences for an item in different stores.
    - Online orders.
    - Seasonality of purchasing of certain items.
    - Properties of items (perishable/non-perishable)
    - Location of stores (urban/rural).
    - Demographics and customer types.
- Extension for DC to stores distribution of items.

Appendix

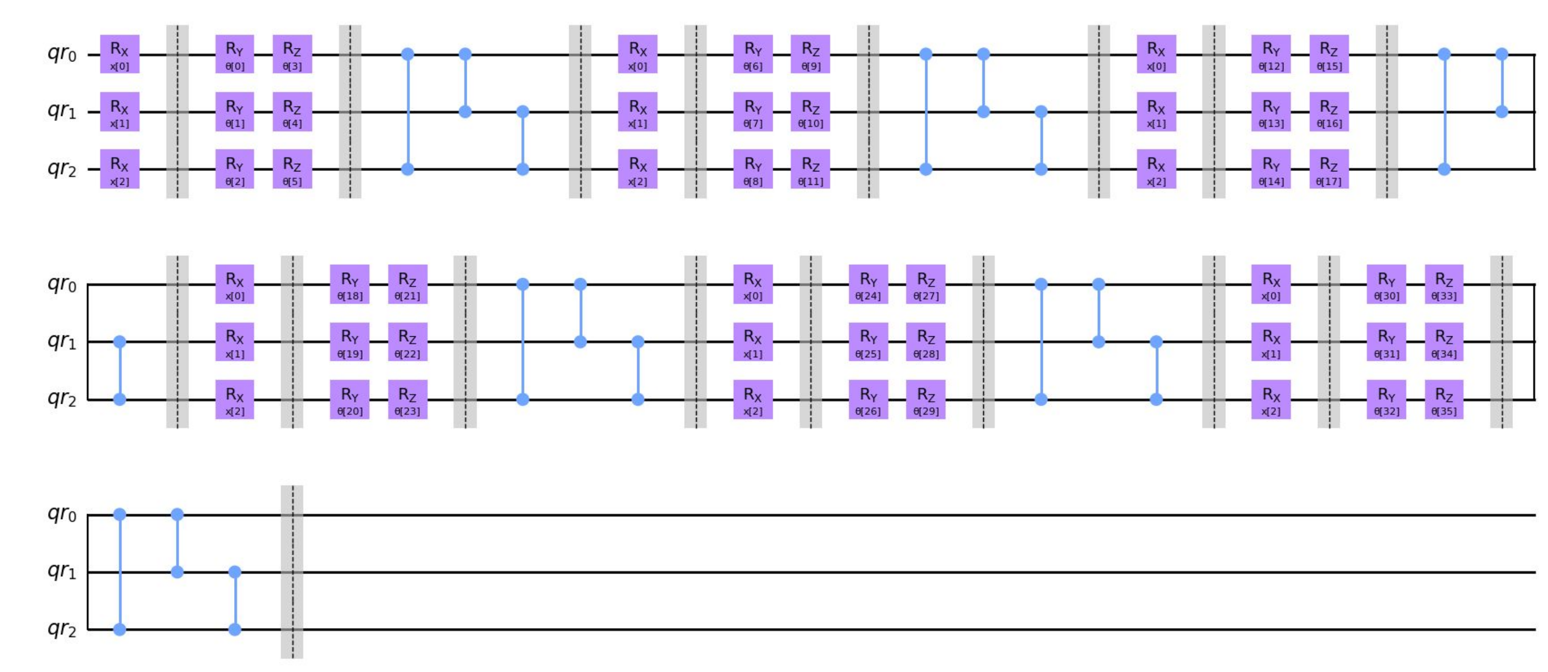
Example – Sales Probability Distribution



$G(x) = (x+1)^2$

$P(x) = G(x)/\text{sum}(G(x))$

Appendix – Parametric Quantum Circuit



Reward Function – Profit per cycle

$$P = min(S, OH + X) \times (P_c + P_r) - X \times (P_c + P_h) - OH \times P_h - max(X - D, 0) \times P_h$$

Where,

- P: Weekly profit.
- D: Demand quantity.
- S: Predicted Sales.
- OH: On-Hand quantity.
- X: Replenishment.
- P_r: Profit per unit for product.
- P_c: Procurement cost per unit for product.
- P_h: Holding cost per unit per week for product.

Replenishment amount

- Model is designed to run for any product and store combination.
- A more thorough evaluation for all product & store combinations to be done later.
- For a specific combination, **replenishment amount is mostly 1 for each week in the 25 weeks forecasting horizon.**
- **Data shows sale of 1 unit on exactly 8 days (in different weeks) from 2020 to 2023.**

.

Produ	StoreI	Date	item_
Item4	Loc14	08-09-2021	1
Item4	Loc14	20-12-2022	1
Item4	Loc14	12-03-2022	1
Item4	Loc14	10-09-2022	1
Item4	Loc14	04-06-2021	1
Item4	Loc14	31-10-2022	1
Item4	Loc14	07-04-2023	1
Item4	Loc14	04-06-2022	1

Safety stock

- Model does not explicitly consider safety stock.
- Sales data indicates **sale of 1 unit on few weeks in 3 years span.**
- Model design **emphasis on reducing on hold cost and ensuring sufficient safety stock as time passes.**

Results – QPU

Week	Capital	On Hand	Demand	Replenishment	Predicted Sale	Actual Sale	Profit
0	1000	1	2	1	1	1	6.6
1	1006.6	1	0	1	0	0	-52.8
2	953.8	2	3	1	3	3	116.6
3	1070.4	0	1	1	1	1	6.6
4	1077	0	0	1	0	0	-52.8
5	1024.2	1	3	1	3	2	61.6
6	1085.8	0	2	1	1	1	6.6
7	1092.4	0	2	1	1	1	6.6
8	1099	0	3	1	1	1	6.6
9	1105.6	0	1	1	1	1	6.6
10	1112.2	0	3	1	2	1	6.6
11	1118.8	0	1	1	1	1	6.6
12	1125.4	0	2	1	2	1	6.6
13	1132	0	2	1	1	1	6.6
14	1138.6	0	0	1	0	0	-52.8
15	1085.8	1	0	1	0	0	-52.8
16	1033	2	0	1	0	0	-52.8
17	980.2	3	2	2	2	2	13.2
18	993.4	3	2	2	2	2	13.2
19	1006.6	3	3	1	3	3	116.6
20	1123.2	1	2	1	2	2	61.6
21	1184.8	0	0	1	0	0	-52.8
22	1132	1	3	1	3	2	61.6
23	1193.6	0	3	1	3	1	6.6
24	1200.2	0	2	1	2	1	6.6

Sample of QPU Costs

Details

IonQ's Aria QPUs are built on a chain of trapped 171Yb+ ions, spatially confined via a microfabricated surface electrode trap within a vacuum chamber. Gates are performed via a two-photon Raman transition using a pair of counter-propagating beams from a mode-locked pulsed laser. This allows for high-quality single and two-qubit transitions and all-to-all connectivity. Initialization is performed via optical pumping, and readout is performed with a combination of a resonant laser, a high numeric aperture lens, and photomultiplier tubes.

IonQ compiles and optimizes your high-level quantum logic gates into the smallest possible set of laser pulses to realize your program on trapped ions, mapping your gates onto ideal pairs for execution using up-to-the minute continuous calibrations.

For single-qubit gates, IonQ uses the GPI gate, the GPI2 gate and the GZ gate. The GPI and GPI2 gates are simply Rabi oscillations made by driving the qubits on resonance using laser beams in a Raman configuration. The GZ gate is performed by advancing/retarding the phase of this laser beam, creating a "virtual" operation.

For entangling, two-qubit gates, IonQ uses the Mølmer-Sørenson gate. This entangling gate and the single-qubit gates above constitute a universal gate set. By irradiating any two ions in the chain with a predesigned set of pulses, it is possible to couple ions' internal states with the chain's normal modes of motion to create entanglement.

Aria also supports error mitigation: error mitigation algorithms aim to reduce systematic errors in noisy quantum systems by splitting a circuit into an ensemble, and post-processing the outputs. IonQ's error mitigation algorithm is called debiasing, where the results are classically post-processed using component-wise averaging or majority voting.

Learn more about this device

Hardware provider
IonQ

Availability
Weekdays, 12:00:00 - 03:00:00 UTC

Pulse control
Not supported

Region
us-east-1

Location
Maryland, USA

Native gate calibrations
Not supported

Cost
- $0.30 / task + $0.03 / shot (on-demand)
- $7000.00 / hour (reservation)

Qubits
25