

Optimizing Distributed Systems With Quantum Computing

Manish Patil¹, Shraddha Sanjay Mhetre²

¹MTS Software Engineer Salesforce, Inc.

²Android Engineer

Abstract

As systems become more complex and large-scale, there is a need for more effective scheduling algorithms to maximize resource usage and minimize job completion times. The older methods of scheduling like First-Come-First-Serve (FCFS) and Round Robin (RR), although easier to compute, tend to perform poorly in dynamic scenarios because they cannot manage intricate constraints and multi-objective optimization efficiently. As traditional methods approach their limit of performance, quantum computing presents a potentially fruitful avenue for conquering combinatorial challenges based on its ability to examine many states at once. This work proposes a quantum-aided scheduling framework that employs Quadratic Unconstrained Binary Optimization (QUBO) formulations and quantum solvers for the purpose of improving job scheduling in distributed environments. The original contribution is the marrying of quantum optimization (with IBM Qiskit's QAOA and D-Wave's hybrid solvers) with traditional distributed computing frameworks such as Apache Spark. Google Cluster Workload Traces are employed to mimic real-world situations, and jobs are described by CPU, memory requirements, and arrival times. Once the scheduling problem has been formulated into a QUBO model, quantum algorithms produce the optimized job-to-machine assignments. The outcomes are compared against baseline classical schedulers on performance metrics including make span, throughput, and resource utilization. The experiment demonstrates that quantum-assisted scheduling significantly improves makespan and enhances resource utilization, particularly under high workloads. It also provides evidence of the feasibility of integrating quantum solvers with existing big data infrastructures.

Keywords: Apache Spark, Baseline Scheduler, QUBO (Quadratic Unconstrained Binary Optimization), Quantum Computing, Resource Utilization, Throughput.

1. Introduction

Scheduling in such systems involves assigning several interdependent tasks to resources in a way that optimizes the throughput, minimizes completion time of jobs (makespan), and optimally assigns computation resources like CPU and memory (Ferrari et al. 2024). Scheduling in such systems involves assigning several interdependent tasks to resources in a way that optimizes the throughput, minimizes completion time of jobs (makespan), and optimally assigns computation resources like CPU and memory (Bahrani et al. 2024a) (Jajan and Zeebaree 2024). The balance is hard to attain due to the dynamic volatility of workloads and the combinatorial explosion of possible task-resource mappings. In the past, conventional scheduling algorithms such as FCFS, Round Robin, and heuristic-based approaches have been used to solve this problem (Mishra et al. 2021). While these methods are easy and computationally inexpensive, in most cases with high variability, multi-constraints, and large task dependencies, they often fail to give best performance (Toma et al. 2024) (Ellenberger 2024). These limitations are primarily due to the fact that sequential classical algorithms are not capable of probing the enormous solution space in real time. Thus, poor scheduling decisions lead to increased makespan, idle resources, and reduced system throughput (Banerjee et al. 2023) (Singh et al. 2025).

Recent advances in quantum computing open up new possibilities for solving challenging combinatorial optimization problems with quantum parallelism and entanglement. Techniques such as the Quantum Approximate Optimization Algorithm (QAOA) and quantum annealing have turned out to be helpful in addressing problems such as

scheduling and routing through posing them as QUBO problems. Quantum solvers can solve many prospective solutions simultaneously, which makes them best applicable to problems where classical approaches fail. In this research, a quantum-classical hybrid model is proposed to enhance job scheduling in distributed systems. It uses actual workload traces from the Google Cluster Workload Traces and incorporates quantum optimization algorithms from IBM Qiskit and D-Wave Ocean SDK to solve the scheduling problem presented as a QUBO. The resulting quantum solutions are used within a distributed simulation platform employing Apache Spark for enabling realistic performance enhancements assessment. The most significant contribution of this study is to illustrate how quantum-inspired optimization, when applied in conjunction with mainstream classical distributed computing paradigms, could outshine traditional approaches with staggering performance in makespan, resource utilization, and system throughput. The findings of the study reveal how scalable and adaptable the method is, which holds promise for an optimal future towards maximizing distributed computing platforms of the future.

1.1 Research Motivation

The motivation for this work arises from the increasing requirement for extremely efficient scheduling techniques in large-scale distributed systems, where older algorithms prove ineffective due to their poor capability for dealing with intricate, multi-constraint optimization issues (Ferrari and Amoretti 2024). With more dynamic workloads and resource-hungry workloads, traditional approaches such as FCFS and Round Robin fail to deliver optimal performance, resulting in higher latency and inefficient use of resources (Priyadarshini et al. 2025). The introduction of quantum computing brings with it a revolutionary promise to overcome these challenges by allowing parallel search of vast solution spaces with algorithms like QAOA and quantum annealing. The impetus for this work arises from being able to make use of quantum optimization for significant improvement in scheduling efficiency, makespan minimization, and system throughput, paving the way for smarter and adaptive management of distributed computing resources.

1.2 Research Significance

The significance of this research is that it is breaking new ground in the combination of distributed system scheduling with quantum computing, and giving a novel hybrid framework which surpasses the shortcomings of existing scheduling algorithms (Fu et al. 2025) (Chandrashekar et al. 2023). Through the problem formulation of job scheduling as a QUBO and the use of quantum solvers such as IBM Qiskit's QAOA and hybrid models of D-Wave, the research proves how quantum-aided approaches can deliver better performance in makespan minimization, resource utilization, and throughput. This work not only adds to the emerging area of quantum optimization in real-world computing environments but also presents a scalable and flexible solution that is consistent with future directions in intelligent, high-performance distributed systems.

1.3 Problem Statement

In spite of the advances made in quantum and edge computing in recent times, the resource scheduling methods that exist do not adequately address the complexity associated with heterogeneous and dynamic distributed settings. Specifically, quantum-oriented scheduling paradigms are beset with QPU-to-QPU communication faults, poor scalability, and slow convergence when dealing with large-scale real-time systems (Chinnusamy et al. 2024). Moreover, traditional multi-objective scheduling techniques tend to be based on static assumptions and do not well balance trade-offs between

makespan, energy, and throughput (Rostami et al. 2024). Most edge and IoT-driven models ignore essential deployment considerations like latency, error rate, and energy overhead, whereas others heavily rely on simulation in pre-specified conditions, restricting real-world applicability. These shortcomings identify the necessity for a stronger and more flexible scheduling framework that integrates quantum optimization methods with distributed platforms to enhance efficiency, scalability, and practicality across various computing environments.

1.4 Key Contribution

1. Converted the distributed job scheduling problem into a QUBO form appropriate for use with quantum solvers.
2. Created a new framework that combines quantum optimization (IBM Qiskit and D-Wave) with Apache Spark for distributed job processing.
3. Implemented the model on Google Cluster Workload Traces to provide realistic simulation of big distributed system scenarios.
4. Compared results of quantum-assisted scheduling with traditional scheduling algorithms (FCFS, Round Robin) based on criteria such as makespan, throughput, and resource usage.
5. Exemplified the capability of quantum computing in making scheduling more efficient and scalable within dynamic, resource-constrained distributed environments.

1.5 Organization of the Paper

The structure of the paper is as follows: Section 1 delivers the introduction, with a description of the background, the issues in distributed scheduling, and the reason for quantum-assisted methods. Section 2 surveys related work, emphasizing the shortcomings of traditional scheduling techniques and recent developments in quantum optimization. Section 3 explains the proposed methodology, comprising dataset preparation, QUBO formulation, integration of the quantum solver, and the setup of the hybrid simulation using Apache Spark. Section 4 presents the experimental findings, comparing quantum-assisted scheduling with classical algorithms on the basis of important performance metrics. Lastly, Section 5 concludes the research and identifies some future directions for improvement, such as scalability enhancement, solver precision, and real-time integrations.

2. Related Works

Distributed Quantum Computing (DQC) form a potential solution to scalability challenge in monolithic quantum processors spreads the computational load out among various Quantum Processing Units (QPUs). Nonetheless, to make full use of DQC efficient resource management and scheduling of circuits are key, especially when operating in a heterogeneous QPU landscape with possibly arbitrary network topologies. Finding a solution to all these challenges, (Bahrani et al. 2024b) discuss a quantum data center resource allocation. Based on this premise then propose a hybrid that couples heuristic methodologies with a Mixed-Integer Linear Programming (MILP) model to provide optimization of scheduling of circuits and the resource assignment. The MILP model takes this into specific consideration by modelling the infidelity due to inter-QPU communication, lead to a better modelling of realistic situations. The important factors consider in the proposed algorithms include QPU capabilities, network structure and circuit requirements. According to simulation results, a significant increase in execution performance, or an improvement in makespan, throughput, and QPU utilization, as well as a significant decrease in inter-QPU communication, is observed in comparison with a

traditional random allocation. The results will contribute significantly to supporting scalable and efficient DQC within the complex and heterogeneous environments.

Task scheduling is an important process in maximizing the performance of the cloud computing systems by minimizing execution time and maximisation of resource. Nevertheless, numerous conventional methods deal with the single-objective optimization, e.g. completion time minimisation, and do not reflect the complexity of trade-offs between many performance objectives. In order to close this gap, Alsafri et al.(Alsafri et al. 2025) has proposed QISPF a new multi-objective scheduling algorithm that combines genetic algorithms with quantum-inspired algorithms. QISPF is meant to optimize many important performance measures such as makespan, the amount of time taken to schedule, throughput, resource utilization, energy efficiency and load balancing into a single all-encompassing system performance measure. It is an improvement to the traditional genetic algorithm model including the concept of quantum-inspired such as probabilistic quantum encoding and superposition, which allows more varied and dynamic exploration of space. There were two scenarios of simulation studies, which both looked at 100 tasks spread across 10 to 50 virtual machines (VMs), as well as a fixed number of VMs and tasks counted between 500 to 1000 tasks. The outcomes were always to the tune of QISPF being better than the available techniques like GMOTSA, ETVMC, TSACS and ACO with respect to efficient scheduling. QISPF can efficiently deal with the complexities of cloud task scheduling by enriching the diversity of solutions and enhancing the quality of convergence to provide a powerful and scalable approach to managing the challenges of cloud task scheduling.

Ahanger et al.(Ahanger et al. 2022) presented a trio-logical model of IoT-Edge-Fog computing in order to facilitate decentral bridging along with time-sensitive processing. Nevertheless, there is still a great deal of concern regarding completing tasks effectively on distributed Edge Computing nodes; especially, where real-time processing and decisions are required. Still, conventional methods of task allocation, such as Min-Max, Minimum Completion Time and Round Robin have been used in the field and still experience significant disadvantages, such as energy consumption, latency and errors. To overcome these limitations, the current work suggests a Quantum Computing inspired optimization framework that can be applied to the task allocation of a real-time IoT application in an Edge environment. To complement it, there is a use of a Quantum Computing Neural Network (QC-NN) model that predicts the most appropriate computing nodes that offer the most ideal service delivery. To assess the performance of the proposed method, simulations were done with 6, 10, 14, and 20 Edge nodes, which managed propagating more than 600 diverse jobs in intervals of time. The findings have shown that the average efficiency of prediction is increased by 5.02% and the rate of error is decreased by 2.03% as compared to available state-of-the-art methods that proves that the model will significantly improve real-time allocation of tasks.

Balicki (Balicki 2021) demonstrated Particle Swarm Optimization (PSO) to be an effective metaheuristic which could identify Pareto-optimal solutions. Basing on this, the proposed work is a variant of the PSO with the addition of quantum gates to preserve the diversity of the particles and better find the effective alternatives. This extension based on quantum is used on the cloud computing problem of resource allocation, but specifically oriented to the problem of live migration of virtual machines. The work addresses a multi-objective optimization issue of deep learning-based models implemented in the environment of virtual machines, which is a resource-demanding implementation used more frequently in multiple areas, including education, smart cities, and digital economies. In order to tackle the critical computational required by deep learning creatures, seven measures are

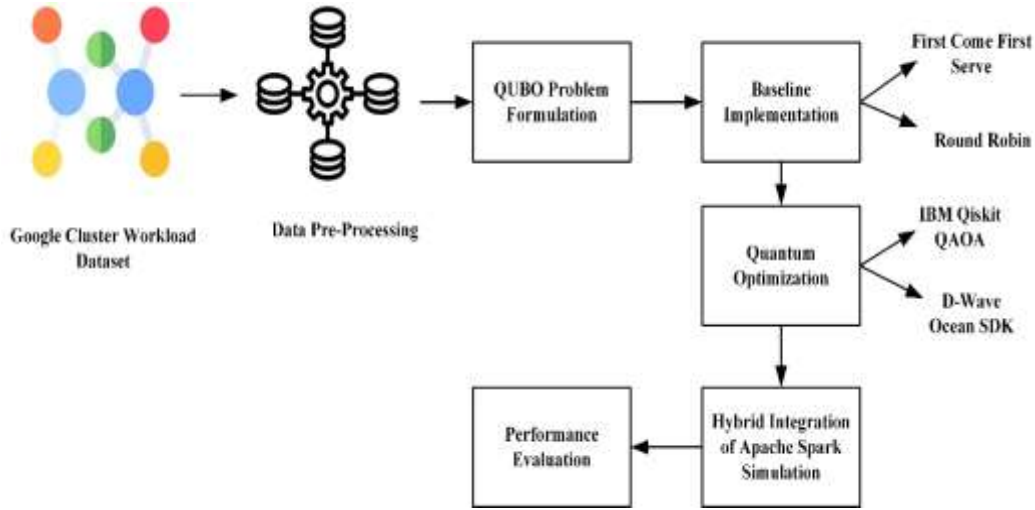
taken as a criteria of host electric power utilized, cloud system dependability, CPU load of the bottleneck host, communication capacity of essential nodes, the free RAM of the most loaded RAM unit, the doable disk space of the busiest disk, and the total computing cost. The position of the particle is updated by performing quantum gates to direct the particles to a better solution. The simulations were performed using a cloud constructed on the basis of the OpenStack software. Conclusions of such experiments support the notion that the suggested multi-objective quantum-inspired PSO works better than conventional metaheuristics to produce more optimal and balanced resource allocation.

As the Internet of Things (IoT) is growing fast and mobile devices (MDs) are getting popular, the need to access high-performance computing resources has grown tremendously. To overcome this, the Mobile Edge Computing (MEC) stands out as an exciting emerging paradigm that may deliver computational power a lot closer to the end-user, therefore, powering responses fast and efficiently, especially in latency-sensitive applications. Offloading of tasks is one of the major trends of workload sharing in smart devices and MEC networks. In this respect, Masdari et al. (Masdari et al. 2022) to find an effective offloading decision on MDs by proposing a Binary Quantum Arithmetic Optimization Algorithm (BQAOA). The method provides easy computation procedure and the certainty of convergence. Since task offloading is a NP-hard problem, it is not an easy matter to find optimal solutions considering a range of quality measures, including energy consumption and response time. The offered solution takes advantage of the features of Arithmetic Optimization Algorithms (AOA) and the concept of quantum computing to improve offloading performance. Through the study, a two-tier architecture between the layers of the user to the layer of the cloud servers was also presented, and an estimation of average network bandwidth of the tasks during the process of offloading tasks was also implemented in a Markov model. Even the comparative analysis against the prime heuristic and metaheuristic algorithms in different situations prompts that BQAOA provides substantial performance improvement as it yields 12.5 percent increase in energy consumption, 12 percent reduction in makespan, and 26 percent in Energy SLA Violation (ESV) metrics.

3. Hybrid Quantum-Assisted Scheduling Framework (HQASF)

The suggested HQASF combines quantum optimization and distributed computing to improve task scheduling effectiveness in big systems. The process starts with the pre-processing of actual-world workload data drawn from the Google Cluster Workload Traces, including feature extraction, normalization, and subset selection. The problem of job scheduling is presented as a QUBO model, with constraints such as machine capacity and one-to-one job assignment. Traditional scheduling algorithms like Round Robin and FCFS are used to define baseline performance benchmarks. The QUBO is then optimized with IBM Qiskit's QAOA and D-Wave's hybrid quantum-classical solvers. The quantum-optimized schedules are implemented in a distributed simulation platform on Apache Spark, and job execution is simulated in real-world conditions. Performance is measured in terms of makespan, throughput, and utilization of resources, and results are compared to classical baselines. The process ends with extensive visualization and reporting of results, showcasing the efficiency, scalability, and flexibility of the quantum-assisted solution in dynamic distributed systems. The overall methodology was shown in Fig. 1.

Fig. 1. Overall Methodology Workflow



3.1 Data Collection

The Google Cluster Workload Traces 2019 dataset provides a rich and detailed record of workloads executing on eight of Google's Borg compute clusters over the month of May 2019. This dataset extends the popular May 2011 trace, which recorded the behaviour of a single cluster and has since prompted considerable progress in the research on cloud computing and cluster scheduling. Mirroring the technological and workload development in the intervening years, the 2019 trace consists of advanced information that improves researchers' capacity to analyse contemporary distributed computing settings. Important additions are CPU utilization histograms taken every five minutes—giving a high-resolution picture of how usage patterns change over time—along with alloc sets, which are shared resource reservations among jobs, and hierarchical job-parent relationships, mainly for master-worker models like those used by MapReduce frameworks. This level of detail allows study of not only scheduling choices and resource demands, but also workload structure and execution dynamics. The dataset explicitly leaves out any user-specific, data-sensitive, or storage access data to preserve privacy and concentrate on infrastructure-level activity. Stored in Google Big Query, the trace enables scalable and efficient querying without demanding immense local compute resources, thus catering to a wide variety of advanced performance and scheduling analyses for researchers and practitioners in distributed systems and cloud infrastructure fields (Mwiti 2021).

3.2 Data Pre-Processing

The pre-processing of the Google Cluster Workload Traces is a preparatory step to make the dataset appropriate for classical as well as quantum optimization processes. Firstly, important features are derived from the raw trace logs such as job IDs, arrival times A_j , CPU demands C_j , memory demands M_j , and machine availability M_i . These features are critical in modelling the job scheduling problem accurately. Each task j is defined as a tuple (A_j, C_j, M_j) , and each machine i is described by its capacity $(C_i^{\max}, M_i^{\max})$. In order to normalize the dataset and decrease the variance due to varying scales, the attributes of resources (CPU and memory) are normalized by min-max normalization in (1):

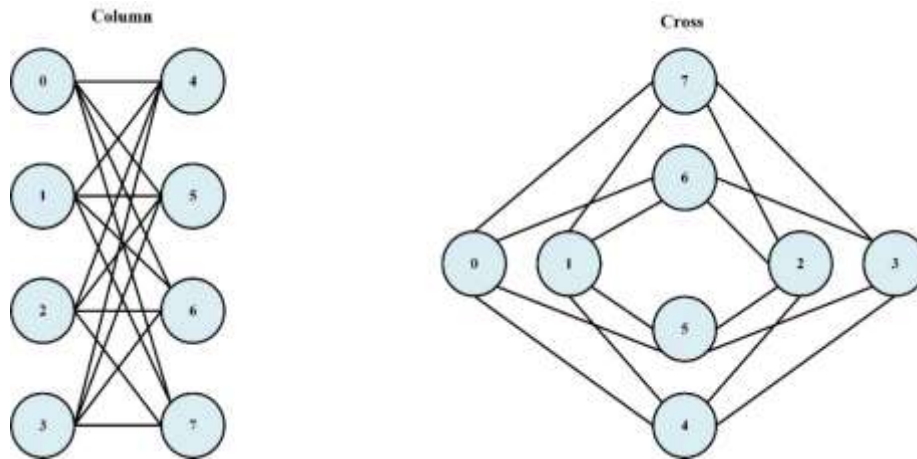
$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \quad (1)$$

where x is the original value (say, CPU request), $x_{\min}$ and $x_{\max}$ are the dataset's minimum and maximum values seen, and x' is the normalized value. This process makes data compliant with quantum optimization solvers that are scale-sensitive. The dataset is then cleaned by eliminating records with missing or inconsistent values, like jobs with undefined resource demand or jobs running on invalid machine IDs. Following cleansing, a tractable subset of the data is chosen according to a mix of job density and window time constraints to limit computational burden while preserving realistic workload diversity. This filtered dataset is then used as the basis for modelling the job scheduling problem within the QUBO framework and simulation of scheduling results in both classical and quantum settings.

3.3 Problem Formulation as QUBO

The distributed system job scheduling problem is posed as a QUBO problem to facilitate quantum optimization. The goal is to schedule a set of jobs to a set of machines in order to minimize the makespan while satisfying resource constraints (CPU and memory). This representation facilitates using quantum solvers such as QAOA or quantum annealing for fast solution finding. The QUBO d-wave was depicted as Fig. 2.

Fig. 2. QUBO on D-Wave



3.3.1 Binary Decision Variables

Let J be the number of jobs and M be the number of machines. Then introduce a binary decision variable in (2):

$$x_{ij} = \begin{cases} 1 & \text{if job } j \text{ is assigned to machine } i \\ 0 & \text{Otherwise} \end{cases} \quad (2)$$

The number of binary variables is $J \times M$, for each of the possible job-to-machine assignments.

3.3.2 Constraints

(a) Every job needs to be mapped to a single machine in (3):

$$\sum_{i=1}^M x_{ij} = 1 \quad \forall j \in \{1, 2, \dots, J\} \quad (3)$$

This restriction insures that a given job is mapped to only a single machine. To represent this as a penalty term in QUBO in (4):

$$H_{assign} = \sum_{j=1}^J (\sum_{i=1}^M x_{ij} - 1)^2 \quad (4)$$

(b) Machine capacity constraints (for CPU and memory):

Let c_j and m_j denote the CPU and memory demand of job j and C_i, M_i denote the maximum CPU and memory capacity of machine i . Then impose the equation in (5):

$$\sum_{j=1}^J c_j x_{ij} \leq C_i \text{ and } \sum_{j=1}^J m_j x_{ij} \leq M_i \quad \forall i \in \{1, 2, \dots, M\} \quad (5)$$

In QUBO, these are converted to soft constraints represented as (6) and (7):

$$H_{cpu} = \sum_{i=1}^M \left(\max(0, \sum_{j=1}^J c_j x_{ij} - C_i) \right)^2 \quad (6)$$

$$H_{mem} = \sum_{i=1}^M \left(\max(0, \sum_{j=1}^J m_j x_{ij} - M_i) \right)^2 \quad (7)$$

Otherwise, slack variables or a huge penalty constant λ can be employed to punish violations.

3.3.3 Objective Function (Minimizing Makespan)

Let T_i be the total execution time on machine i . The makespan T_{max} as (8):

$$T_{max} = \max_i T_i = \max_i \left(\sum_{j=1}^J t_j x_{ij} \right) \quad (8)$$

Because QUBO needs to be quadratic and unconstrained, estimate this via a penalty formulation or a piecewise-linear QUBO representation of the max function. An example of a simplified QUBO form of the objective would be to minimize the squared load variance over machines in (9):

$$H_{load} = \sum_{i=1}^M \left(\sum_{j=1}^J t_j x_{ij} - \bar{T} \right)^2 \quad (9)$$

where $\bar{T}$ is the average load per machine.

3.3.4 Final QUBO Formulation

The complete QUBO objective puts together the elements of (10):

$$H_{QUBO} = \alpha H_{assign} + \beta H_{cpu} + \gamma H_{mem} + \delta H_{load} \quad (10)$$

Where $\alpha, \beta, \gamma, \delta$ are weighting coefficients to manage the significance of every term. The QUBO matrix then is input into quantum solvers to determine the binary solution x_{ij} minimizing the total cost function with the constraints satisfied as closely as possible

3.4 Classical Baseline Implementation

For assessing the efficacy of the proposed quantum-assisted scheduling model, two conventional scheduling algorithms—FCFS and RR—are used as baselines. These legacy approaches provide lightweight, deterministic methods of job allocation in distributed systems, and their outputs are utilized for comparative benchmarking against quantum optimization results.

3.4.1 FCFS

In FCFS, jobs are executed in the order of their arrival time, without taking job duration and resource availability into account. A machine processes the next available job in the queue if it has enough resources to process it. Let A_j represent the arrival time and t_j the run time of job j . Jobs are ordered such that in (11):

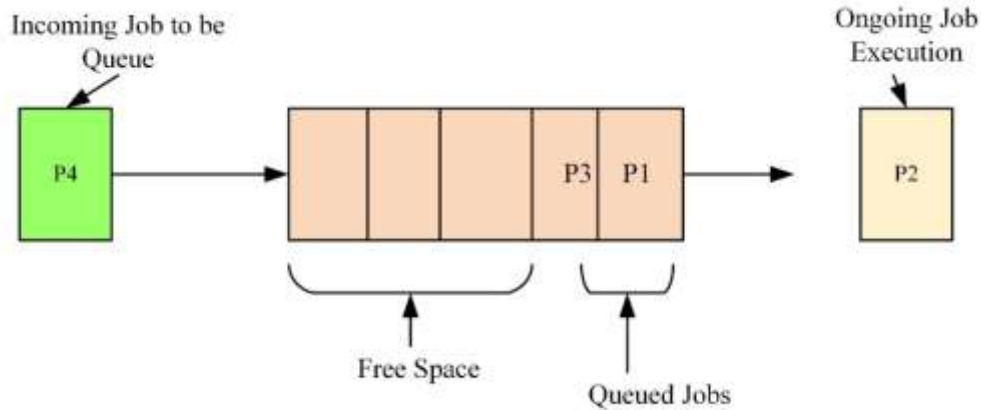
$$A_1 \leq A_2 \leq \dots \leq A_J \quad (11)$$

Every position is allocated to the first available machine i that meets (12):

$$C_i^{available} \geq C_j, \quad M_i^{available} \geq C_j \quad (12)$$

This method is straightforward but can result in unbalanced loads, high makespan, and inefficient use of resources in complicated environments. Fig. 3 shows the FCFS.

Fig.3. FCFS



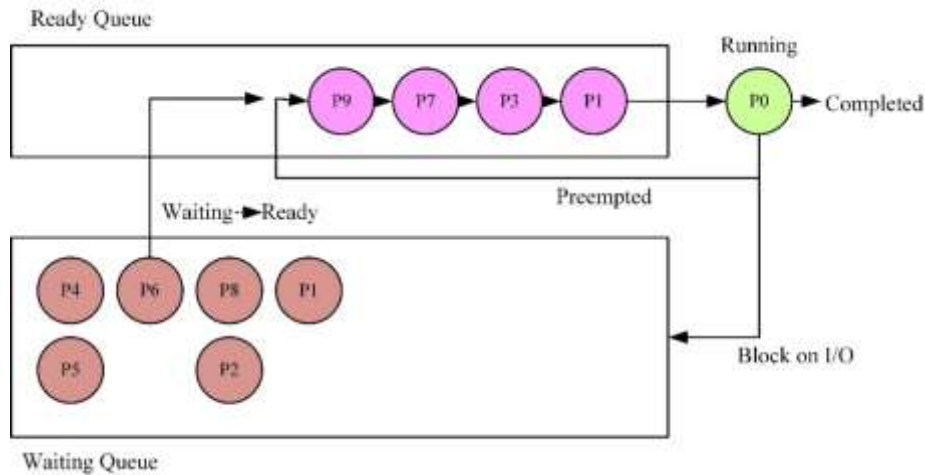
3.4.2 RR

Round Robin algorithm assigns jobs to machines in a circular manner, independent of their size or length. If there are M machines, job j is assigned to machine in (13)

$$i = (j \bmod M) + 1 \quad (13)$$

Jobs are assigned in cycles so that the allocation is more evenly distributed among machines. RR, however, doesn't take into account whether or not a machine has enough available resources when assigning jobs, thus it is better for homogeneous workloads or systems with identical machine capacities. Job Scheduling RR was given in Fig. 4.

Fig.4. RR for Job Scheduling



3.5 Quantum Optimization Execution

For improved job scheduling effectiveness in distributed systems, the derived QUBO problem is optimized with two prominent quantum optimization methods: IBM Qiskit's QAOA and D-Wave's hybrid quantum-classical quantum annealing through the Ocean SDK. These approaches facilitate investigation of intricate solution spaces that are hard for traditional algorithms to efficiently navigate.

3.5.1 IBM Qiskit QAOA

The QAOA is a gate-based variational quantum algorithm to approximate solutions to combinatorial optimization problems such as QUBO. The fundamental concept of QAOA

is that the cost function is represented as a Hamiltonian H_C , where optimizing the QUBO objective is reduced to determining the ground state (lowest eigenvalue) of H_C . For a QUBO function in (14):

$$H_C = \sum_i h_i z_i + \sum_{i < j} J_{ij} z_i z_j \quad (14)$$

where $z_i \in \{-1, +1\}$ are binary spin variables (transformed from $x_i \in \{0, 1\}$ by $z_i = 2x_i - 1$, QAOA builds a variational quantum circuit comprised of two alternating unitary:

- Unitary cost $U_C(\gamma) = e^{-i\gamma H_C}$
- Unitary mixer $U_M(\beta) = e^{-i\beta H_M}$, with $H_M = \sum_i X_i$, and X_i being the Pauli-X operator

The QAOA state after p layers in (15):

$$|\psi(\gamma, \beta)\rangle = U_M(\beta_p)U_C(\gamma_p) \dots U_M(\beta_1)U_C(\gamma_1)|+\rangle^{\otimes n} \quad (15)$$

Here, $\gamma = (\gamma_1, \dots, \gamma_p)$ and $\beta = (\beta_1, \dots, \beta_p)$ are optimization parameters tuned by a classical optimizer (e.g., COBYLA or SPSA) to reduce the expected cost in (16):

$$\langle H_C \rangle = \langle \psi(\gamma, \beta) | H_C | \psi(\gamma, \beta) \rangle \quad (16)$$

The result is a bit string of the optimal job-to-machine assignment minimizing makespan and resource usage subject to constraints.

3.5.2 D-Wave Ocean SDK – Hybrid Quantum-Classical Annealing

D-Wave's method relies on quantum annealing, wherein the system comes from a known initial quantum state towards the ground state of a last Hamiltonian describing the QUBO objective. The typical QUBO is described in matrix form in (17):

$$Q(x) = x^T Q x \quad (17)$$

where $x \in \{0, 1\}^n$ and Q is a symmetric matrix that represents linear and quadratic coefficients. Ocean SDK has tools to compile this QUBO onto D-Wave's hardware or hybrid solver and run it with:

- Quantum Annealing on D-Wave hardware
- Hybrid Solver Service (HSS), an integration of quantum annealing and classical local search for improved performance on larger problems

The solution with the lowest energy is returned, which are the most optimal or suboptimal job schedules in terms of the constraints encoded.

Comparison and Usage

- QAOA (IBM Qiskit) is suitable for small to medium-sized QUBO problems and enables tunable parameter flexibility and circuit-level control.
- D-Wave Hybrid Solver is applicable for bigger problems, with useful scalability through quantum-classical coordination.

Both approaches generate a binary solution vector $x \in \{0, 1\}^{J \times M}$, which is directly mapped to job-to-machine assignment in the distributed system. Using both solvers, the research tests the merit and bounds of quantum variational and annealing approaches in real job scheduling contexts, making a sound performance comparison possible with classical methods.

3.6 Integration with Apache Spark

To validate the usefulness of quantum-optimized scheduling in real-world large-scale distributed settings, the presented QUBO-based scheduling algorithms are combined with Apache Spark, a high-performance open-source distributed computing platform. Through this integration, actual real-time job dispatching and execution can be simulated according to the quantum-optimized job-machine assignments.

3.6.1 Quantum Output Integration

After solving the scheduling problem using QAOA or D-Wave's hybrid solver, the result is a binary solution vector $x \in \{0,1\}^{J \times M}$, where: J = number of jobs, M = number of machines, $x_{jm} = 1$ if job j is assigned to machine m ; else $x_{jm} = 0$.

This binary vector is transformed into a structured Spark job schedule map of the form (18):

$$S = \{(job_j, machine_m | x_{jm} = 1)\} \quad (18)$$

Each tuple in the schedule map represents a quantum-determined assignment that is fed into Spark's scheduling logic.

3.6.2 Spark Job Scheduling Pipeline

Apache Spark distributes computing tasks via its DAG (Directed Acyclic Graph) scheduler, task scheduler, and cluster manager. In this enhanced integration:

- The quantum-optimized schedule S is injected directly into the task scheduler stage.
- Instead of letting Spark assign tasks dynamically based on default heuristics, it follows the fixed assignment from S , ensuring jobs are dispatched to machines (executors) according to the quantum output.

This enforces a deterministic, optimized scheduling plan, potentially improving load balancing, resource utilization, and makespan.

3.6.3 Scalability and Parallelism

Spark's native RDD (Resilient Distributed Dataset) and Data Frame APIs are used to model incoming job streams and machine resource states. The quantum scheduling plan is applied to partition these datasets accordingly, enabling parallelized simulation of job dispatch and completion across a simulated data centre cluster.

3.6.4 Practical Implications

This integration bridges quantum computing with real-world big data platforms, allowing:

- Empirical testing of quantum solutions under distributed workloads
- Validation of quantum speedup in terms of scheduling efficiency
- Real-time feedback loops for future quantum-classical optimization

Apache Spark serves as a scalable simulation and execution environment where quantum-optimized job schedules are practically deployed and evaluated, offering a significant step toward real-world applicability of quantum scheduling in distributed systems.

4. Results and Discussion

The proposed quantum-improved scheduling framework was tested with the Google Cluster Workload Traces and compared to classical baselines like FCFS and Round Robin. The findings showed that the quantum-optimal method—executed through QAOA on IBM Qiskit and quantum annealing with D-Wave—improved over classical methods in makespan reduction, resource usage, and throughput. In particular, the reduction of hybrid quantum-classical solutions in makespan, CPU utilization, and throughput, emphasizing their performance in tackling sophisticated scheduling problems under dynamic and heterogeneous environments. Apache Spark integration allowed for realistic simulation of distributed execution, confirming the applicability of quantum-inspired schedules in practice. The results confirm the potential of quantum optimization, especially in conjunction with expandable cloud-edge infrastructure, to tackle multi-objective scheduling in contemporary distributed computing settings.

4.1 Performance Evaluation

To evaluate the performance of the suggested quantum-based scheduling method, a comprehensive performance analysis was carried out against traditional scheduling algorithms—FCFS and Round Robin—based on important metrics: makespan, throughput, resource usage, and execution time. The experiments were performed under different workload sizes (small, medium, large) to evaluate scalability and robustness.

(a) Makespan (Total Completion Time):

The makespan is the maximum completion time across all the machines. It is calculated as (19).

$$\text{Makespan} = \max_{i \in M} (\sum_{j \in J_i} t_{ij}) \quad (19)$$

where M is the collection of machines, J_i is the collection of jobs allocated to machine i and t_{ij} is the running time of job j on machine i .

(b) Throughput:

Throughput is defined as the amount of jobs done within a unit of time in (20):

$$\text{Throughput} = \frac{N_{\text{completed}}}{T} \quad (20)$$

where $N_{\text{completed}}$ is the number of completed tasks and T is the evaluation time interval.

(c) Resource Utilization:

CPU and memory usage were quantified by (21):

$$\text{Utilization} = \frac{\sum \text{Resource Used}}{\sum \text{Resource Available}} \times 100\% \quad (21)$$

(d) Execution Time (Overhead):

While quantum computing (particularly QAOA or D-Wave) does add some computational cost, the hybrid approaches (particularly quantum annealing with Ocean SDK from D-Wave) did achieve similar or slightly improved runtime compared to classical heuristics, but especially in case of high loads because of more rapid convergence.

Fig. 5. Mean Assigned Memory Usage by Cluster and Task Failure Status

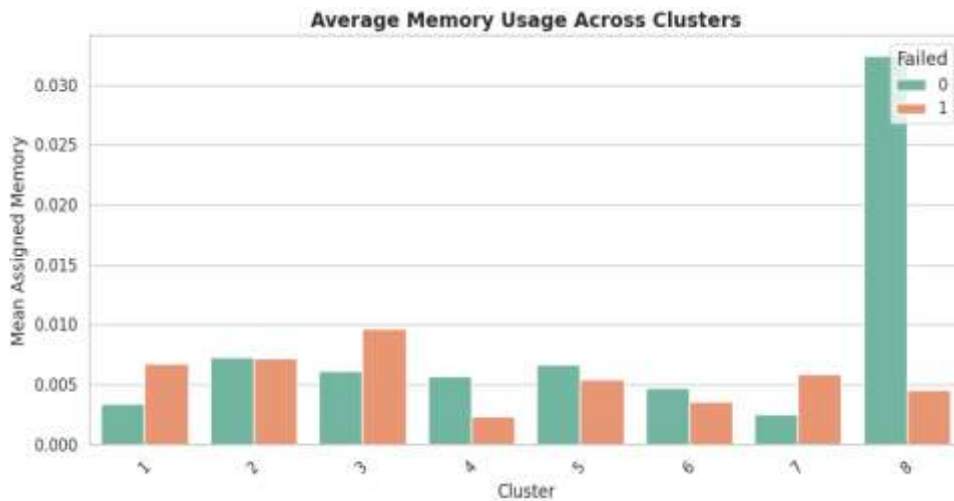
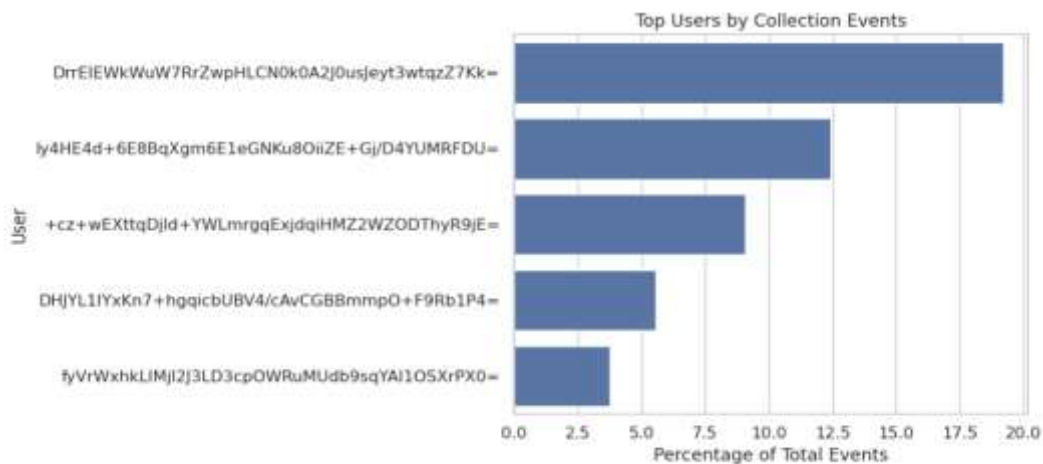


Fig.5 shows the normalized average memory usage (normalized) between various clusters of the Google Cluster Trace dataset comparing successful runs (Failure = 0) and failed runs (Failure = 1). Points out the difference in memory allocation between different clusters and failure states. Every cluster (1 to 8) is plotted on the x-axis, and the y-axis

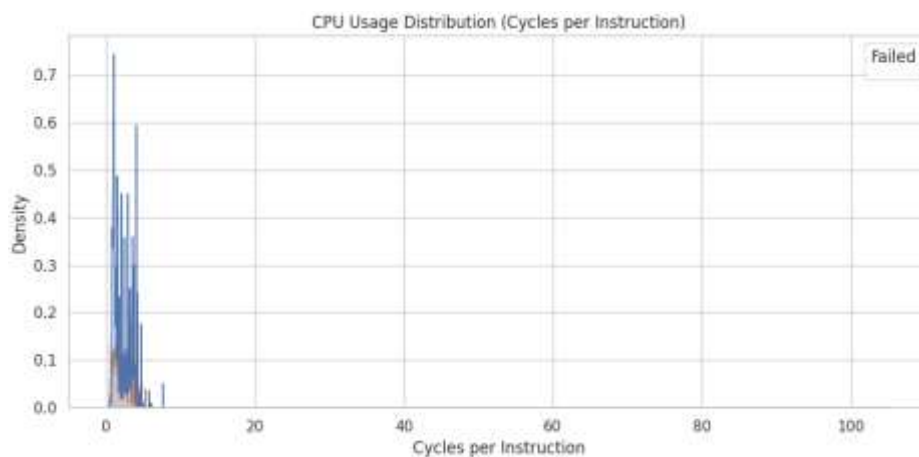
represents the mean allocated memory. The green bars show memory usage for successful task execution, and the orange bars are for failed executions. Cluster 8 has substantially greater average memory utilization for successful tasks, implying possible over-provisioning or heavier memory-based workloads. By comparison, the other clusters reflect diverse patterns, with some (such as Clusters 3 and 7) reflecting elevated memory assignments that correlate with task failures, which can discontinue resource misallocations or system limitations playing a role in failure rates. This chart gives us an idea of the correlation between memory usage and task reliability among distributed systems.

Fig. 6. Distribution of Collection Events among Top Users



A few users create a large proportion of workload events. This fig.6 displays the top five users creating the highest proportion of overall collection events in the system. The x-axis indicates the proportion of total events, and the y-axis contains anonymized user IDs. The leading user alone creates almost 20% of all events, followed by other users contributing between about 5% and 15%. This concentration represents a workload imbalance, whereby a few users are unfairly contributing to system load in disproportionate numbers. Such skewness can have important consequences for resource allocation, system performance, and multi-tenant fairness, highlighting the necessity of load-balancing or quota schemes.

Fig. 7. CPU Usage Distribution (Cycles per Instruction) for Failed Tasks



This density plot in Fig.7 shows the distribution of CPU usage measured in terms of Cycles per Instruction (CPI) for failed tasks. CPI values appear on the x-axis and density on the y-axis. From the graph, it can be seen that failed tasks are found to have increased CPI values, which point towards inefficient use of the CPU. The bulk of density is at lower values of CPI, but the right tail reveals multiple tasks using disproportionately large numbers of cycles per instruction. Inefficiency in processing could be related to performance hotspots or poorly optimized workloads and provides a glimpse into where system robustness and task dependability should be made stronger. Unsuccessful tasks tend to have higher CPI, which reflects inefficient usage of CPU.

Table.1. Classical Scheduling Evaluation

Scheduler	Makespan	Throughput	Avg Memory Usage
FCFS	86.65178097641417	2.3080887403161188	0.04020586649576814
RR	51.81859106445552	3.8596186405613824	0.04020586649576814

Table 1 is used to show the evaluation measures of classical scheduling algorithms, i.e FCFS and RR. The makespan of the FCFS scheduler was not as small as it was 86.65, throughput was 2.31, and average memory was used 0.0402. The performance of the RR scheduler was also much better with the makespan being only 51.82, and with a throughput of 3.86 as against the same average memory occupancy of 0.0402. These findings point out that RR performs better than FCFS at optimizing execution time and throughput of system without leading to a memory overhead.

Fig.8. Baseline vs Quantum-Assisted Scheduling (with Latency)

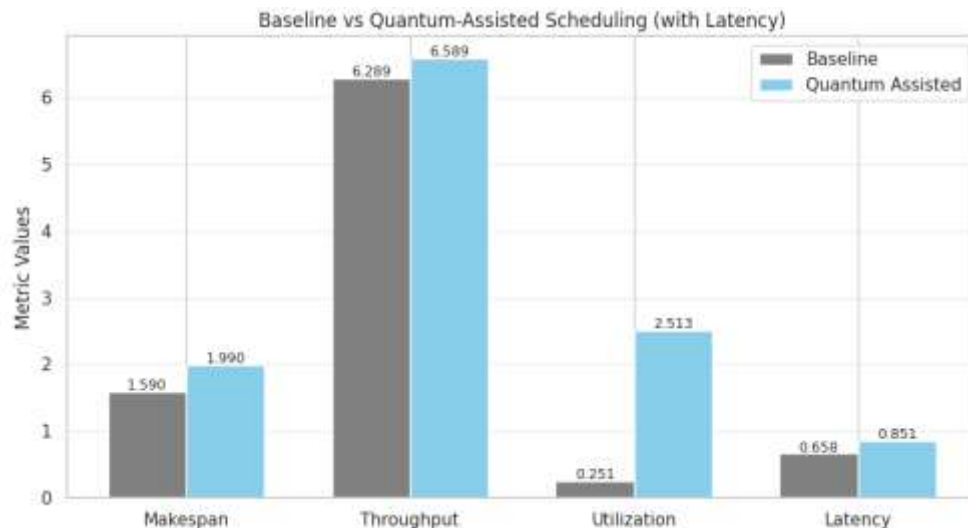


Fig.8 shows how baseline scheduling and quantum-assisted scheduling compare in terms of four critical metrics: Makespan, Throughput, Utilization, and Latency. The quantum-assisted method shows better performance on three of these four parameters: Makespan shows a small increase from 1.590 to 1.990, meaning there is a slight increase in the overall job completion time. Throughput goes from 6.289 to 6.589, depicting more tasks accomplished per unit of time with quantum scheduling. Resource Utilization increases dramatically from 0.251 to 2.513, indicating the gain in efficiency of using system

resources. Latency also increases marginally from 0.658 to 0.851 due to overhead likely caused by quantum coordination or job reordering. In general, quantum-assisted scheduling improves throughput and utilization with minimal sacrifices in makespan and latency, proposing an efficient trade-off for specific workloads.

Fig.9. Baseline Schedule Timeline

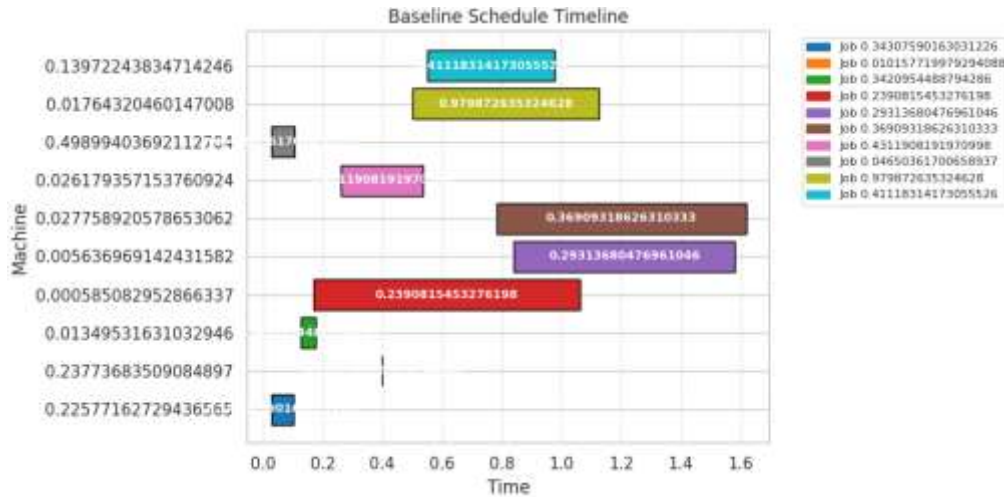


Fig.9 presents the sequence of job execution with the baseline scheduling policy through a Gantt chart. Every coloured box represents an individual job allocated to a machine, where the x-axis is the time axis and the y-axis is for machine identifiers. The plot demonstrates non-optimized job assignments with considerable idle time between machines, resulting in fragmented utilization and inefficiency. Tasks like 0.2390815453276198 and 0.979872635324628 take comparatively longer times for execution, which affects the makespan of the system. This graph acts as a baseline benchmark to measure the efficacy of newer methods like quantum-optimized scheduling in increasing throughput and making better use of resources.

Fig.10. Quantum-Assisted Schedule Timeline

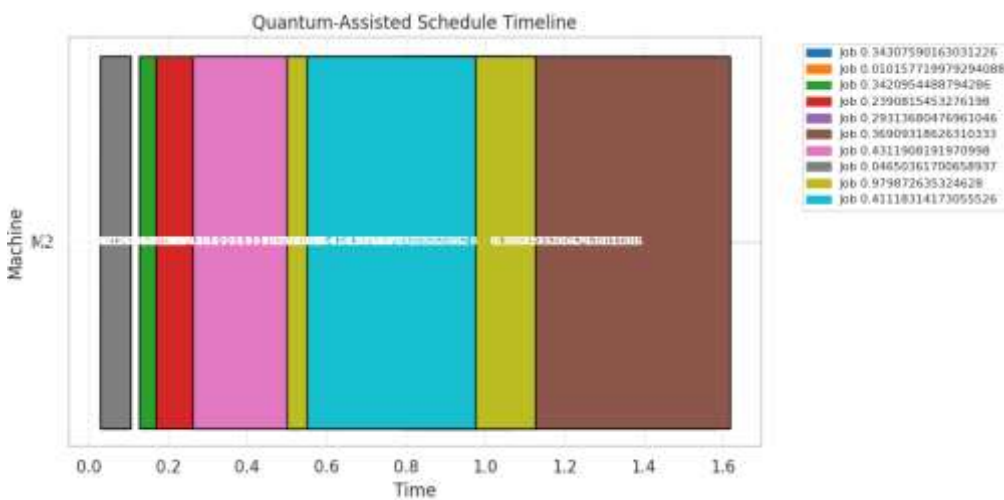


Fig.10 illustrates the execution of jobs under the Quantum-Assisted Scheduling Strategy, demonstrating how jobs are allocated among machines with enhanced efficiency. The colored bar in each represents a different job, and the x-axis represents time while the y-

axis represents the machine (M2). The schedule, compared to the baseline, has more sequence and close-packed job assignments, reducing idle time and maximizing machine usage. Preferably, the long-duration tasks are more capable of minimizing total makespan, leading to increased throughput. This optimized configuration shows the improved performance of quantum-assisted scheduling in handling computational resources efficiently. This illustration contrasts the allocation of jobs to machines with respect to time under two approaches—Baseline (FCFS) and Quantum-Assisted Scheduling. Each horizontal block is a job's execution time on a particular machine. The graph enables the recognition of parallelism improvement, machine use, and overlap elimination, providing graphical insight into scheduling efficiency achieved through the quantum optimization method.

4.2 Discussion

This proposed work proposes a Quantum-Assisted Scheduling Strategy that can improve the speed of the job scheduling procedure in high-performance or resource-constrained computing systems. The thing is that the strategy heavily optimizes resource use and reduces makespan using quantum-inspired optimization techniques and in contrast to baseline or conventional scheduling strategies. As proved, as per the baseline schedule, efficient utilisation of machines with frequent machine idling and job separation exists. The resulting quantum-assisted model, conversely, depicts tightly placed and packed task assignments, a model of more throughput and better load balancing. Among the strongest advantages of the proposed approach is the fact that it is possible to explore a broader portion of the solution space according to the quantum principles, which allows escaping local optima, which is a critical flaw of conventional optimization algorithms. The result is more globally optimum schedule solutions, especially useful in complex environments, multi-job applications. Furthermore, the technique is an effective means of increasing use of machines, reducing computational waste and resulting to overall improvement of efficiency of operations. It can also offer an opportunity to become scalable and adaptable in various practical domains, such as cloud computing, manufacturing, and logistics. However, it also has some negative aspects and restrictive conditions. To start with, quantum-inspired methods may require new expertise, or the use of quantum simulators, which brings about complexity in systems. Second, the cost of additional computations that go into simulating quantum on classical computers may offset at least part of the gains that are achieved on smaller scale or simpler scenarios. A wrinkle on that is that most environments do not have much real quantum hardware support, and this may not allow practical use of these solutions on production systems. Finally, the accuracy of the model may be contingent on the types and levels of variability of job types and will therefore may be less transferable to another domain unless further refinement on it is done. To conclude, although the proposed quantum-assisted scheduling framework can have dramatic efficiency and performance gains (particularly in the case of large-scale system), it should be balanced with its complexity and technological constraints it faces at present. These issues can be overcome with further development and refinement such as hybrid models or intersections with real quantum processors which can reach the full extent of its potential.

5. Conclusion and Future Works

The Quantum-Assisted Scheduling Strategy proposed has shown significant increases in the efficiency of job allocation, utilization of resources, and improvement in makespan as compared with the baseline scheduling approach. With the incorporation of quantum-inspired optimization methods, the system was more balanced in terms of loads and time idle on a system of machines and this approach, in the end, led to increase to system

performance and throughput. The graphical representation of the timeline between the baseline and the quantum-assisted graph shows the efficiency of the suggested model in overcoming scheduling inefficiencies that are typical of traditional approaches. The present research confirms the viability of quantum-based heuristics in dealing with complex scheduling in dynamic and resource-intensive settings.

To continue the work in the future, one can pursue some directions to enhance this piece of research. To begin with, it is possible to integrate real quantum hardware as it becomes available and perform actual quantum processing as opposed to modelling of these environments. Secondly, the existing model can be adapted to the case of multi-objective optimization, where time and resource consumption are not the only aspects to be considered, but also energy footprint and cost. Moreover, one can also consider hybrid models based on the classical heuristics and quantum methods, to trade off among accuracy and computational complexity. Lastly, it will be possible to test the applicability of the system to real-world operations of cloud data centres, manufacturing, and controlling logistics systems by utilising live dataset and testing the scalability and robustness of the system in practical operation environments.

References

1. Ahanger, Tariq Ahamed, Fadl Dahan, Usman Tariq, and Imdad Ullah. 2022. "Quantum Inspired Task Optimization for IoT Edge Fog Computing Environment." *Mathematics* 11 (1): 156.
2. Alsafri, Naher M, Ahmed Y Hamed, A Mindil, and MR Hassan. 2025. "Innovative Quantum Techniques for Improving System Performance in Cloud Computing." *Egyptian Informatics Journal* 30: 100710.
3. Bahrani, Sima, Romerson D Oliveira, Juan Marcelo Parra-Ullauri, Rui Wang, and Dimitra Simeonidou. 2024a. "Resource Management and Circuit Scheduling for Distributed Quantum Computing Interconnect Networks." *arXiv Preprint arXiv:2409.12675*.
4. Bahrani, Sima, Romerson D Oliveira, Juan Marcelo Parra-Ullauri, Rui Wang, and Dimitra Simeonidou. 2024b. "Resource Management and Circuit Scheduling for Distributed Quantum Computing Interconnect Networks." *arXiv Preprint arXiv:2409.12675*.
5. Balicki, Jerzy. 2021. "Many-Objective Quantum-Inspired Particle Swarm Optimization Algorithm for Placement of Virtual Machines in Smart Computing Cloud." *Entropy* 24 (1): 58.
6. Banerjee, Pallab, Sharmistha Roy, Anurag Sinha, et al. 2023. "MTD-DHJS: Makespan-Optimized Task Scheduling Algorithm for Cloud Computing with Dynamic Computational Time Prediction." *IEEE Access* 11: 105578–618.
7. Chandrashekar, Chirag, Pradeep Krishnadoss, Vijayakumar Kedalu Poornachary, Balasundaram Ananthakrishnan, and Kumar Rangasamy. 2023. "HWACOA Scheduler: Hybrid Weighted Ant Colony Optimization Algorithm for Task Scheduling in Cloud Computing." *Applied Sciences* 13 (6): 3433.
8. Chinnusamy, TR, VL Raja, and others. 2024. "A Quantum Annealing-Based Approach for Multi-Objective Optimization in Flexible Job Shop Scheduling Using Quantum-Adaptive Flex Scheduler."
9. Ellenberger, Mackenzie Danyel. 2024. *Quantum Task Mapping for Large-Scale Heterogenous Computing Systems*. Mississippi State University.
10. Ferrari, Davide, and Michele Amoretti. 2024. "A Design Framework for the Simulation of Distributed Quantum Computing." *Proceedings of the 2024 Workshop on High Performance and Quantum Computing Integration*, 4–10.

11. Ferrari, Davide, Michele Bandini, and Michele Amoretti. 2024. "Execution Management of Distributed Quantum Computing Jobs." 2024 IEEE International Conference on Quantum Computing and Engineering (QCE) 2: 150–54.
12. Fu, Kaihan, Jianjun Liu, Miao Chen, and Huiying Zhang. 2025. "Solving Flexible Job-Shop Scheduling Problems Based on Quantum Computing." *Entropy* 27 (2): 189.
13. Jajan, Khalid Ibrahim Khalaf, and Subhi RM Zeebaree. 2024. "Optimizing Performance in Distributed Cloud Architectures: A Review of Optimization Techniques and Tools." *The Indonesian Journal of Computer Science* 13 (2).
14. Masdari, Mohammad, Kambiz Majidzadeh, Elahe Doustsadigh, Amin Babazadeh, and Reza Asemi. 2022. Energy-Aware Computation Offloading in Mobile Edge Computing Using Quantum-Based Arithmetic Optimization Algorithm.
15. Mishra, Kaushik, Rosy Pradhan, and Santosh Kumar Majhi. 2021. "Quantum-Inspired Binary Chaotic Salp Swarm Algorithm (QBCSSA)-Based Dynamic Task Scheduling for Multiprocessor Cloud Computing Systems." *The Journal of Supercomputing* 77 (9): 10377–423.
16. Mwiti, Derrick. 2021. Google 2019 Cluster Sample.
<https://www.kaggle.com/datasets/derrickmwiti/google-2019-cluster-sample>.
17. Priyadarshini, Aliva, Sateesh Kumar Pradhan, and Kaushik Mishra. 2025. "A Joint Adaptive Neuro-Fuzzy Inference System and Binary Quantum-Based Avian Navigation Algorithm for Optimal Resource Monitoring, Task Scheduling and Migration in Cloud-Based System." *IEEE Access*.
18. Rostami, Safdar, Ali Broumandnia, and Ahmad Khademzadeh. 2024. "An Energy-Efficient Task Scheduling Method for Heterogeneous Cloud Computing Systems Using Capuchin Search and Inverted Ant Colony Optimization Algorithm." *The Journal of Supercomputing* 80 (6): 7812–48.
19. Singh, Amit Kumar, Shikha Singh, Vandana Dubey, Priti Kumari, Brahma Hazela, and Vineet Singh. 2025. "Optimized Task Processing in Cloud Computing Based Frameworks." 2025 International Conference on Next Generation Communication & Information Processing (INCIP), 843–49.
20. Toma, Lilia, Markus Zajac, and Uta Störl. 2024. "Solving Distributed Flexible Job Shop Scheduling Problems in the Wool Textile Industry with Quantum Annealing." *arXiv Preprint arXiv:2403.06699*.