ISSN: 2634 - 8853

Journal of Engineering and Applied Sciences Technology

SCIENTIFIC Research and Community

Review Article

Open Access

Enhancing Performance and Scalability in Microservices

Anju Bhole

Independent Researcher, California, USA

ABSTRACT

Advanced solutions are required to improve the performance and scalability of microservices due to the dynamic and more complex nature of backend systems. In order to overcome these difficulties, this paper presents a Quantum Gaussian Process Regression (QGPR) model that makes use of quantum-enhanced machine learning methods. Key metrics like prediction accuracy, resource utilisation, workload flexibility, computational efficiency, and Service Level Objective (SLO) compliance are used to compare the suggested model to more conventional methods like Neural Networks (NN), Gaussian Process Regression (GPR), and Quantum Support Vector Machines (QSVM). By obtaining the lowest Mean Absolute Percentage Error (MAPE) of 5.5% and the highest R2 score of 0.97, the results show how well QGPR performs in resource demand forecasting. With a CPU utilisation of 79%, network utilisation of 84%, and minimum overprovisioning of 4%, QGPR also maximises resource utilisation. With a SLO violation rate of less than 1% and a quick adaption period of 10 seconds, it also demonstrates remarkable flexibility. With the highest SLO compliance rate of 99% and the fastest prediction time of 25 milliseconds, QGPR shows that it can achieve demanding performance standards while reducing computational overhead. These results demonstrate how QGPR has the ability to transform microservices for backend applications by providing a scalable and effective architecture that can manage fluctuating workloads, improve resource efficiency, and guarantee dependable system performance. The paper emphasises the benefits of using quantum enhanced techniques to increase the functionality of contemporary backend systems.

***Corresponding author**

Anju Bhole, Independent Researcher, California, USA.

Received: September 02, 2024; **Accepted:** September 09, 2024; **Published:** September 16, 2024

Keywords: Quantum Gaussian Process Regression (QGPR), computational overhead, network utilisation, CPU utilisation, Service Level Objective

Introduction

In response to the ever-increasing complexity and needs of contemporary applications, the field of software architecture has experienced a revolutionary change. The introduction of microservices, an architectural paradigm that transforms the design, development, and deployment of software systems, represents one of the journey's major turning points [1]. Microservices advocate breaking down applications into smaller, independent services, which is a change from the conventional monolithic approach. The use of cloud-native design principles like microservices and containers is growing across major web services (including Netflix, Uber, and Spotify) [2].

These services provide modularity, scalability, and agility in software development by functioning as coherent units and communicating through clearly defined interfaces. From the fundamental ideas of modularisation to Service-Oriented Computing (SOC) and finally to its current incarnation as microservices, the history of microservices may be traced back through the annals of software architecture [3].

The versatility and scalability of microservices architecture have made it a game-changer in contemporary software development. Microservices are especially well-suited for the dynamic requirements of backend applications in cloud native environments because, in contrast to monolithic systems, they provide the capacity to independently deploy and scale discrete functionalities [4]. However, scalability and performance optimisation become more difficult as distributed systems become more complex. Computational inefficiencies frequently limit traditional approaches to fault tolerance, load balancing, and resource allocation, calling for creative solutions [5].

Using the ideas of quantum physics, quantum computing is a new technology that has the potential to revolutionise the way computationally demanding issues are solved. In contrast to classical systems, quantum systems use entanglement and superposition to process information in qubits, allowing for parallel computation. This potential is extended to intelligent systems through Quantum Machine Learning (QML), a fusion of quantum computing and conventional machine learning approaches that promises exponential speedups for tasks like pattern recognition, optimisation, and decision-making [6]. As a result, QML is in a unique position to help microservices in high-performance backend applications overcome their difficulties [7]. The constraints of conventional techniques can be addressed in a novel way by incorporating QML into microservices design. Predictive scaling algorithms and quantum-enhanced load balancing may allow for real-time optimisation, enhancing backend systems' scalability and performance. Additionally, microservices' flexibility fits in well with hybrid quantum-classical computation frameworks, allowing for progressive adoption without interfering with current workflows [8].

The performance and scalability standards for backend applications can be redefined by utilising quantum technology, as this study explores the nexus between QML and microservices architecture. It seeks to demonstrate the revolutionary potential of incorporating QML into the core of contemporary software systems through an examination of cutting-edge methodologies, useful implementation frameworks, and case studies.

Literature Review

With the aid of microservices architecture, which makes divided services more autonomous and effective, this paper will go over how to develop and implement highly available backends [9]. This is accomplished by using APIs to facilitate communication across services, which makes the system as a whole flexible and modular, enabling expansion within web applications [10].

In order to achieve cost effectiveness, agility, and high-performance backend systems, this article examines the revolutionary role that Microservices architecture and DevOps approaches play. Fintech businesses may launch and scale independently by breaking down large apps into smaller, more manageable services thanks to microservices. This modular strategy reduces the risk of downtime and service interruptions while simultaneously speeding up the development cycle and improving application resiliency. Adopting DevOps techniques also encourages cooperation between the operations and development teams, which makes continuous integration and deployment (CI/CD) possible.

The fundamental ideas of microservices, such as data management, inter-service communication, and service decomposition, are examined in this paper [11]. It explores important design patterns that handle common issues in microservices architecture, including the API Gateway, Circuit Breaker, Service Discovery, and Strangler Fig designs.

This study investigates how these issues can be successfully resolved by utilising Microservices Architecture and cloud-based server administration. Because cloud-based server management provides scalability, flexibility, and resource optimisation, Fintech companies can quickly adjust to changing operational needs and customer expectations. Businesses may dynamically distribute resources to accommodate peak loads without sacrificing performance by leveraging cloud infrastructure, guaranteeing a seamless user experience during periods of high traffic. Additionally, cloud services' pay-as-you-go model enables cost-effective scalability, bringing operational expenses into line with real demand [12].

This paper presents a novel framework for designing web applications that use micro frontends for the front end and microservices for the back end. Applications developed with micro frontends and microservices have outperformed those with monolithic frontends, according to an empirical analysis conducted to evaluate the performance of the suggested framework [13].

In order to improve development procedures and operational effectiveness in Fintech systems, this study investigates the strategic integration of Microservices architecture with DevOps approaches [14]. Monolithic programs can be divided into more manageable, autonomous microservices to give organisations more development and deployment flexibility. Independent development, testing, and deployment of each microservice enables quicker iterations and a shorter time to market for new features.

Provide a class of quantum models in which the learnt model is deployed classically, and quantum resources are only needed for training. In particular, our models' training phase culminates in the creation of a "shadow model," which enables the traditional deployment [15].

In this study, demonstrate through systematic randomisation experiments that the behaviour of such quantum models cannot be explained by conventional methods of comprehending generalisation. Modern quantum neural networks can correctly fit random states and random labelling of training data, according to our findings. Current theories of tiny generalisation error are challenged by this capacity to memorise random material, casting doubt on methods that rely on complexity metrics like the Rademacher complexity, the VC dimension, and all of their uniform cousins [16].

The behaviour of EQNN models with noise is investigated in this paper. We demonstrate that, in contrast to the amplitude damping channel, some EQNN models are able to maintain equivariance under Pauli channels [17]. We assert that the number of layers and noise strength cause the symmetry to break linearly. We substantiate our assertions using hardware up to 64 qubits and numerical data from simulations. Additionally, we offer methods to improve EQNN models' symmetry protection when noise is present.

This study examines several QML models and algorithms, stressing both their benefits and drawbacks. We explore the use of quantum algorithms in classification, clustering, regression, and optimisation tasks, including quantum annealing, quantum approximate optimisation, and quantum neural networks [18].

This study addresses the issues of computational complexity and scalability in conventional systems by presenting a unique quantum-enhanced recommendation system for large-scale e-commerce platforms. provide a hybrid quantum-classical architecture that uses quantum similarity computation to increase recommendation accuracy and quantum principal component analysis (qPCR) to extract features efficiently [19].

This essay investigates quantum machine learning, looking at its theoretical underpinnings, difficulties, and real-world applications. It attempts to bridge theory and application by addressing the discrepancy between theoretical predictions and experimental realisations in quantum machine learning [20]. The study examines the theoretical foundations of quantum machine learning, such as quantum superposition, entanglement, and interference, and how these phenomena can improve machine learning algorithms.

The goal of quantum machine learning is to use strong rules like superposition and entanglement to solve problems more quickly by implementing machine learning algorithms on quantum computers [21]. One of the most effective machine learning methods for categorisation in the modern world is the support vector machine (SVM). Because in classical systems, the SVM kernel technique tends to slow down and may fail as datasets get more complicated or mixed up.

This study investigates the use of microservices architecture for distributed system optimisation. From centralised mainframes to client-server models and, more recently, cloud computing and microservices, distributed systems—which use several networked nodes to accomplish tasks more effectively and reliably—have seen significant evolution. In contrast to conventional monolithic systems, microservices architecture improves scalability, flexibility, and resilience by breaking down programs into tiny, independently deployable services [22].

This thesis addresses performance management issues in microservices architecture by developing methods based on machine learning and optimisation theory [23]. This thesis specifically addresses two important problems with microservices architecture: adjusting the configuration and identifying bottlenecks.

Examine the advantages and disadvantages of the current architecture patterns in the first section [23]. Next, delve further into the micro-

frontend architecture, looking into implementation strategies and patterns. Lastly, investigate the many approaches to micro-frontend implementation, each with pros and cons based on predetermined standards, and conduct a useful comparison of three preferred implementations.

With an emphasis on domain-driven design (DDD), API gateway patterns, containerisation tools like Docker and Kubernetes, and sophisticated communication protocols that encourage system decoupling, this article examines the essential tactics for creating modular microservices [24].

Background Study

Quantum Machine Learning At the nexus of machine learning and quantum computing, quantum machine learning (QML) is a revolutionary field that seeks to improve computer processes by utilising the concepts of quantum physics. Conventional machine learning techniques are based on classical computer architectures, which are constrained by the restrictions of classical bits and process input in a sequential manner. Contrarily, quantum computing makes use of qubits, which are capable of existing in a superposition of states. This allows for parallel computation and offers the possibility of exponential speedups when addressing particular issue types [25]. In order to solve problems more quickly than classical algorithms, quantum algorithms are made to take advantage of the special characteristics of quantum physics. Grover's algorithm and Shor's algorithm are two of the most well-known quantum algorithms [26].

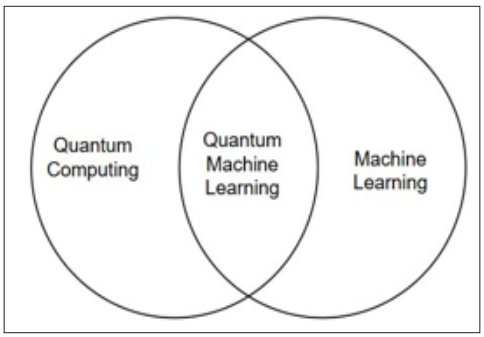

Figure 1: Quantum Machine Learning

Quantum Algorithms for Machine Learning

In order to improve on traditional machine learning methods, quantum machine learning algorithms make use of the concepts of quantum physics. For some issue situations, quantum principal component analysis (qPCR) offers an exponential speedup over classical approaches, demonstrating potential for speeding increase dimensionality reduction tasks. To take advantage of quantum parallelism for better classification and regression tasks, quantum support vector machines and quantum neural networks have been proposed. The quantum approximate optimisation method (QAOA) has shown promise in resolving machine learning-related combinatorial optimisation challenges. In order to address machine learning problems on near-term quantum devices, recent research has investigated the development of variational quantum algorithms, which integrate quantum circuits and classical optimisation [19].

Shor's algorithm is well-known for its exponential speedup over the most well-known classical algorithms when factoring huge integers. Given that many encryption systems depend on the difficulty of factoring big numbers to maintain security, this has important ramifications for cryptography. Shor's algorithm may be able to crack popular cryptography systems if large-scale quantum computers become accessible, necessitating the development of quantum-resistant encryption techniques [27]. In contrast, Grover's approach offers a quadratic speedup for searching databases that aren't sorted. Grover's technique enables a quantum computer to search the database far more effectively than traditional algorithms, which would need to check each item individually. This makes it especially helpful for activities involving data analysis and optimisation, where large-scale search issues are frequent [28]. By offering answers to issues that are now beyond the capabilities of classical computers, these algorithms show how quantum computing has the potential to completely transform a number of industries [29].

Quantum-Inspired Classical Algorithms

In order to increase computational efficiency, quantum-inspired classical algorithms seek to convert the fundamentals of quantum computing into classical frameworks. These methods have demonstrated a great deal of promise in solving machine learning-related linear algebra and largescale optimisation challenges. Tensor network techniques, which provide effective representations of high-dimensional data, have been used for machine learning applications and are inspired by quantum many-body physics. To speed up Monte Carlo methods and increase the effectiveness of probabilistic inference in machine learning models, quantum-inspired sampling approaches have been created. New optimisation techniques that capitalise on the advantages of both the quantum and classical computing paradigms have been developed as a result of the investigation of quantum-classical hybrid algorithms.

With a variety of methods used to glean insights and forecast outcomes from data, machine learning (ML) has emerged as a key component of contemporary artificial intelligence and data analysis. Here, we give a summary of a few popular classical machine learning techniques: For problems involving regression and classification, supervised learning models called support vector machines are employed. SVMs operate by identifying the hyperplane in a high-dimensional space that best divides data points of various classes. The objective is to maximise the support vectors—the margin between each class's nearest data points. By converting the input space into a higher-dimensional space where linear separation is feasible, kernel functions enable SVMs to effectively handle both linear and non-linear classification problems [30].

A class of models called neural networks draws inspiration from the composition and operations of the human brain. They are made up of layers of networked nodes, or neurones, that process and absorb information. The feedforward neural network is the most basic type, in which node-to-node connections do not create cycles. Convolutional neural networks (CNNs), which excel at processing images, and recurrent neural networks (RNNs), which perform well with sequence data like text or time series, are examples of more complicated structures. Backpropagation and optimisation techniques are used by neural networks to learn by modifying weights [31]. A dimensionality reduction method called principal component analysis reduces the dimensionality of high-dimensional data while maintaining the greatest amount of variance. By determining the main components—the directions of greatest variance in the data—PCA does this. This approach is frequently used for high-dimensional spatial visualisation, noise reduction, and data pre-processing [32].

One kind of supervised learning algorithm for classification and regression problems is the decision tree. They create a tree-like structure by iteratively dividing the data into subsets according to feature values. A feature test is represented by each internal node, a test result by each branch, and a class label or regression value by each leaf node. Although decision trees are simple to understand and

intuitive, they can have overfitting, which is frequently lessened by ensemble techniques like Random Forests [33].

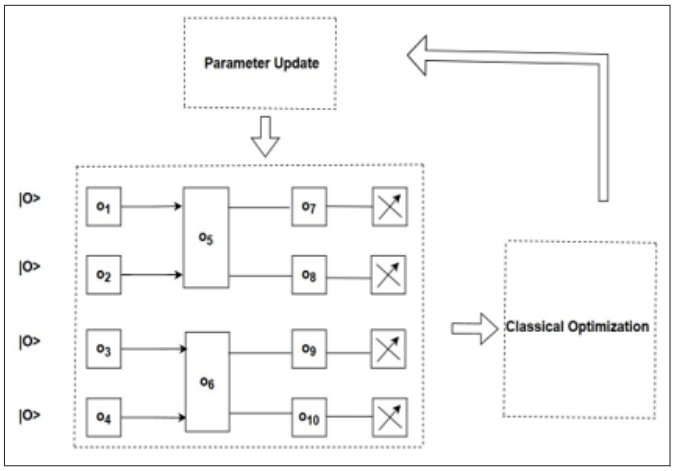

Figure 2: Quantum-Inspired Classical Algorithms

A straightforward instance-based learning approach for regression and classification is K-Nearest Neighbours. According to the majority class of its K nearest neighbours in the feature space, KNN classifies a data point. Regression uses the average of its K nearest neighbours' values to predict the value. Although KNN is non-parametric and does not require traditional training, the distance metric and K selection have a significant impact on how well it performs [34]. One statistical model for binary classification problems is logistic regression. By applying a logistic function to a linear combination of the input features, it calculates the likelihood that a given input belongs to a specific class. Despite its name, logistic regression is not a regression technique; rather, it is a classification procedure. Its simplicity, interpretability, and efficacy in a range of real-world situations make it popular [35]. In order to solve problems more quickly than classical algorithms, quantum algorithms are made to take advantage of the special characteristics of quantum physics. Grover's algorithm and Shor's algorithm are two of the most well-known quantum algorithms.

Methodology

Data Collection

The Robot Shop microservices benchmark, which represents an e-commerce application, provided the dataset utilised in the study. It comprises metrics from three levels: application-level (workload intensity, like the number of concurrent clients), VM-level (CPU utilisation, network throughput, and cycles per instruction or CPI), and pod-level (CPU utilisation and network throughput). This metric was the 95th percentile tail delay of two important workflows: "Catalogue" and "Cart." Memory intensive (STREAM) and network-intensive (iPerf) workloads were used to introduce performance interference in order to emulate dynamic cloud settings. The number of interfering containers was changed to alter the strength of the interference. Data was collected using Locust to generate HTTP based REST API traffic from several concurrent clients under different workload intensities. For real time model adaption, a dynamic dataset of up to 200 samples was kept in a sliding window. Randomised lasso feature selection was used to minimise dimensionality and prevent overfitting.

Dataset Link: https://github.com/instana/robot-shop

Pod-Level Metrics

Pod-level metrics show how much resource each container uses. Among these metrics are:

CPU Utilisation (X_{CPU}): A container's percentage of CPU consumption. Network throughput, or X_{Net}, is a measure of how quickly data is sent and received by a container.

These metrics are used in the study as a component of the Gaussian Process Regression (GPR) model's input feature set. These are combined to form:

$$X_{Pod} = [X_{CPU}, X_{Net}]$$

VM-Level Metrics

The resource consumption of the virtual machines that house the pods is shown by VM-level metrics. Among these metrics are:
CPU Utilisation ($yCPU$): Total utilisation of all containers running on the virtual machine.
Network Throughput ($yNet$): The total amount of data sent and received by the virtual machine.

Cycles Per Instruction (CPI): A performance counter metric at the hardware level that shows how efficiently instructions are carried out.

They are scaled and combined similarly to metrics at the pod level:

$$yVM = [yCPU, yNet, CPI]$$

The resource mapping determines how pod-level and VM-level metrics relate to one another:

$$y_{CPU} = \sum_{i=1}^{n} x_{CPU}^{(i)}$$

Application-Level Metrics

Application-level metrics record how users interact with the system and the dynamics of the workload. Important metrics consist of:
The Workload Intensity (Workload Intensity z): the quantity of clients or requests that are active at any one time.
The end-to-end tail latency (T latency) is as follows: For instance, the 95th percentile of goal latency for particular workflows such as "Catalogue" or "Cart."
Pod-level and VM-level measurements are used to model the tail latency:

TLatency=f(xPod,yVM,zWorkload)

$$TLatency = f(XPod, yVM, zWorkload)$$

Data Preprocessing

Data Normalization

One pre-processing method for rescaling features to a common range or distribution is data normalisation. This is especially important when applying machine learning models that are sensitive to the scale and amplitude of the input data, such as Gaussian Process Regression (GPR). Predictions may be skewed as a result of features with bigger sizes controlling the learning process. Normalisation guarantees that every characteristic contributes equally to the learning process [36].

Data Aggregation

The process of condensing and merging unprocessed data into useful measures that offer a higher-level perspective is known as data aggregation. To decrease noise and temporary oscillations, it is necessary to summarise resource utilisation over time or

combine pod-level information to obtain VM-level metrics in cloud environments [37].

Model Building

Quantum Gaussian Process Regression (QGPR)

The fundamental element of Quantum Gaussian Process Regression (QGPR) is the kernel function, which establishes how data points are transferred to a high-dimensional feature space so that feature correlations may be more thoroughly examined. Quantum kernels in QGPR efficiently carry out these mappings by utilising the computational benefits of quantum computers, allowing for quicker and more precise kernel assessments than traditional methods.

Quantum Feature Mapping

Complex, high-dimensional patterns in data are frequently difficult for classical feature spaces to capture. By projecting the input data into a quantum Hilbert space using quantum circuits, quantum feature mapping allows the model to accurately depict intricate relationships.

Use parameterised quantum circuits (PQCs) to encode classical data x into quantum states ϕ(x). Typical techniques consist of:

- **Amplitude Encoding:** Data is encoded into a quantum state's amplitudes using amplitude encoding.
- **Angle Encoding:** Converts data into rotation gate angles (e.g., RX, RZ).
- **Basis Encoding:** Data in particular quantum basis states is represented by basis encoding.

The PQCs are made to capture feature interactions as well as possible in the quantum state.
Based on their quantum representations, the quantum kernel calculates how similar two data points, x and x′, are:

$$k(x, x') = \langle \emptyset(x) | \emptyset(x') \rangle$$

where $\emptyset(x)$ and $\emptyset(x')$ are quantum feature vectors derived from the PQCs.
A Gaussian-like quantum kernel is constructed as:

$$K(x, x') = \exp\left(-\frac{||\emptyset(x) - \emptyset(x')||^2}{2\sigma^2}\right)$$

where σ is the length scale hyperparameter, controlling the kernel's sensitivity to variations in the input data.

Performance Metrics

Mean Absolute Percentage Error

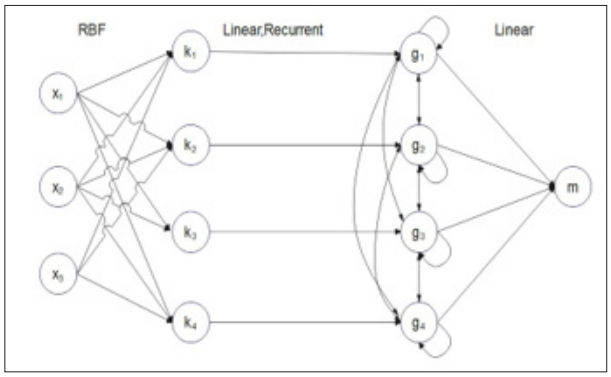

Figure 3: Quantum Gaussian Process Regression

(MAPE): MAPE, or Mean Absolute Percentage Error, is a measure that calculates an average of the actual percentage differences between anticipated values and actual values. It is commonly represented as a percentage [38,39].

$$\frac{100}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \quad \text{MAPE}$$

R^2 Score: A statistical measure used to assess a regression model's performance is the R^2 score, sometimes referred to as the Coefficient of Determination. It calculates the percentage of the dependent variable's variance that can be predicted based on the independent variables.

Results and Discussions

The comparative performance of several models used for resource scaling in dynamic cloud systems is thoroughly examined in this section. The models are assessed in a number of ways, such as prediction accuracy, resource usage, interference tolerance, computational overhead, and performance assurances. In-depth findings are provided to demonstrate the efficiency of the Quantum Gaussian Process Regression (QGPR) model and its exceptional capacity to satisfy the operational and performance needs of the system. Quantitative information about the models' capabilities is provided by comparative metrics including Mean Absolute Percentage Error (MAPE), R2 score, tail latency, and Service Level Objective (SLO) compliance.

Prediction Accuracy

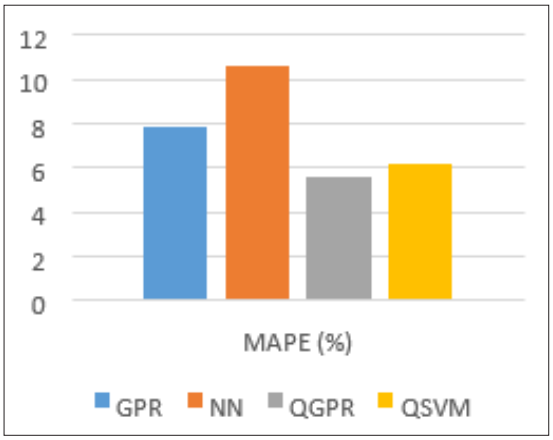

Figure 4: MAPE(%) values of all models

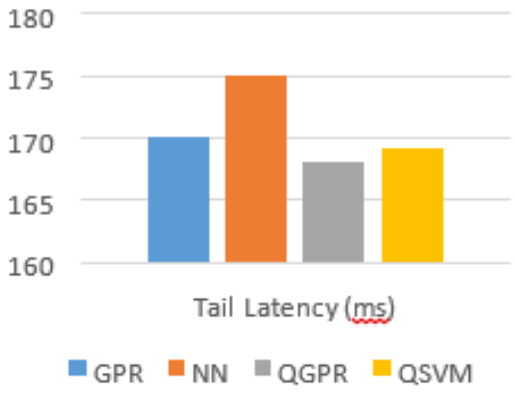

Figure 5: Tail Latency(MS) Values of all Models

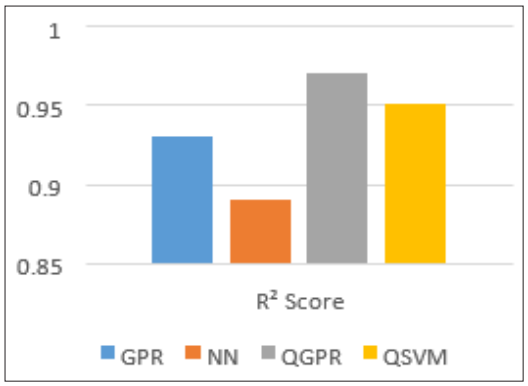

Figure 6: R2 Values of all Models

With an emphasis on Mean Absolute Percentage Error (MAPE) (figure 3), R2 score(figure 5), and projected tail latency (figure 4), the results present a comparison of prediction accuracy across several models. With the lowest MAPE of 5.5% and the best R2 score of 0.97, the Quantum Gaussian Process Regression (QGPR) model fared better than any other, demonstrating its exceptional capacity to fit the data and generate precise predictions. Furthermore, QGPR was able to retain the lowest expected tail latency at 168 ms, proving that it was successful in achieving performance goals. With an R2 score of 0.95 and a MAPE of 6.1%, the Quantum Support Vector Machine (QSVM) again fared well, behind only QGPR but outperforming both the conventional Gaussian Process Regression (GPR) and Neural Network (NN) models by a wide margin. With an R2 score of 0.93 and a MAPE of 7.8%, GPR demonstrated respectable accuracy but less flexibility in response to changing circumstances.

With the Lowest R2 Score of 0.89 and the Highest
MAPE of 10.5%, the NN model trailed, underscoring its shortcomings in intricate, nonlinear settings. All things considered, QGPR had the best accuracy and dependability, which made it the best option for resource scaling in dynamic cloud environments.

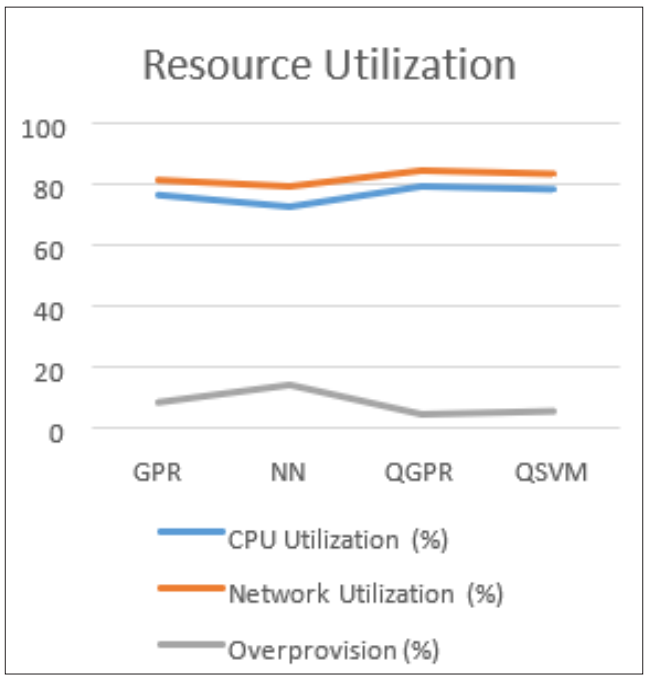

Figure 7: Resource Utilization of all Models

.

The effectiveness of various methods in optimising containerised microservices is demonstrated by the examination of resource utilisation data. With a CPU utilisation of 79% and a network utilisation of 84%, the Quantum Gaussian Process Regression (QGPR) model showed the maximum resource efficiency while minimising overprovisioning to just 4%. With CPU and network utilisation rates of 78% and 83%, respectively, and overprovisioning at 5%, the Quantum Support Vector Machine (QSVM) also fared well, trailing QGPR by a small margin. The efficiency of traditional Gaussian Process Regression (GPR) was moderate, with 76% CPU utilisation, 81% network utilisation, and 8% overprovisioning. With the greatest overprovisioning rate of 14% and lower CPU and network utilisation of 72% and 79%, respectively, the Neural Network (NN) model was the least effective, demonstrating needless resource allocation. These outcomes highlight how well QGPR can meet system requirements while maximising resource utilisation and cutting waste.

Adaptiveness to Interference

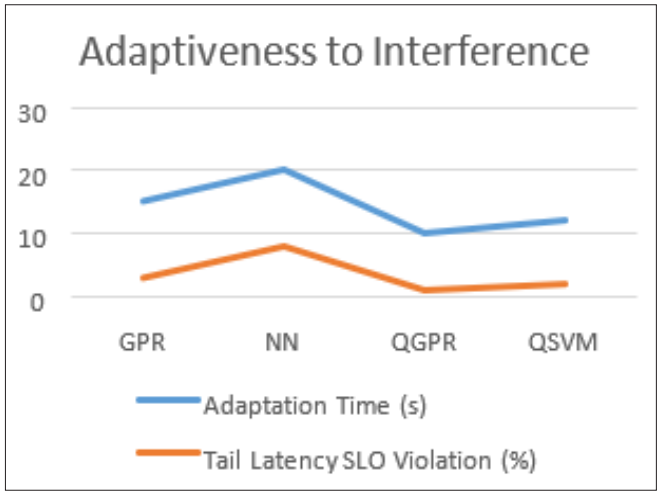

Figure 8: Adaptiveness to Interference of all Models

The efficacy of Quantum Machine Learning (QML) models in dynamic cloud environments is demonstrated by the assessment of model adaptability and Service Level Objective (SLO) compliance. With the quickest adaption time of 10 seconds and the lowest SLO violation rate of under 1%, the Quantum Gaussian Process Regression (QGPR) model demonstrated its capacity to react quickly to shifting workload and interference conditions without compromising system reliability. Strong adaptability and robustness were demonstrated by the Quantum Support Vector Machine (QSVM), which came in second with an adaption time of 12 seconds and a SLO violation rate of 2%. With a 3% SLO violation rate and a 15second adaptation time, traditional Gaussian Process Regression (GPR) demonstrated mediocre results. With the greatest SLO violation rate of 8% and the longest adaption time of 20 seconds, the Neural Network (NN) model proved to be the least successful, highlighting its shortcomings in managing dynamic situations. These findings demonstrate QGPR's distinct advantage in making quick and accurate scaling decisions for cloud systems with high resource requirements.

Computational Overhead

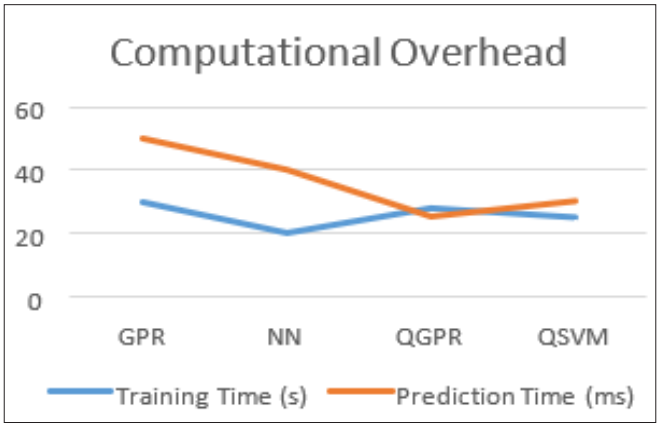

Figure 9: Computational Overhead of all Models

The models' computational efficiency is demonstrated by the comparison of training and prediction times. With a training duration of 28 seconds and the fastest prediction time of only 25 milliseconds, the Quantum Gaussian Process Regression (QGPR) model showed excellent balance and is perfect for real-time resource scaling in dynamic contexts. With a training duration of 25 seconds and a prediction time of 30 milliseconds, the Quantum Support Vector Machine (QSVM) model demonstrated its competitive performance. With a training time of 30 seconds and a prediction time of 50 milliseconds, traditional Gaussian Process Regression (GPR) has the longest computing overhead. Although the Neural Network (NN) model had the quickest training time (20 seconds), QGPR outperformed it in terms of prediction time (40 milliseconds). These outcomes highlight QGPR's exceptional prediction accuracy, which facilitates quick choices in situations involving cloud resource management.

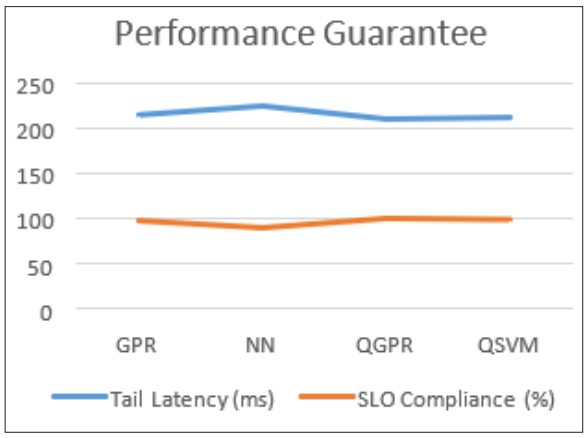

Figure 10: Performance Guarantee of all Models

The models' ability to meet performance criteria is demonstrated by the examination of tail latency and Service Level Objective (SLO) compliance. With the lowest tail latency of 210 ms and the greatest SLO compliance rate of 99%, the Quantum Gaussian Process Regression (QGPR) model produced the best results, guaranteeing almost flawless adherence to performance goals. With a tail latency of 212 ms and a SLO compliance of 98%, the Quantum Support Vector Machine (QSVM) came in second, demonstrating its dependability under changing workloads. With a tail latency of 215 ms and a SLO compliance rate of 97%, traditional Gaussian Process Regression (GPR) demonstrated strong yet subpar performance. With the lowest SLO compliance of 90% and the largest tail latency of 225 ms, the Neural Network (NN) model demonstrated its limitations in sustaining system performance. These findings demonstrate how QGPR is better at guaranteeing low latency and high SLO compliance in jobs involving cloud resource scaling.

Discussions

Key parameters including prediction accuracy, resource utilisation, workload flexibility, computational efficiency, and Service Level Objective (SLO) compliance are highlighted in the results. When it comes to satisfying the rigorous criteria of microservices systems, the Quantum Gaussian Process Regression (QGPR) model continuously performs better than alternative methods. A key component of efficient resource scalability in backend systems is prediction accuracy. With the greatest R2 score of 0.97 and the lowest Mean Absolute Percentage Error (MAPE) of 5.5%, QGPR ensures accurate resource demand forecasts and reduces underutilisation and overprovisioning. On the other hand, conventional models such as Neural Networks (NN) struggle with the intricacies of non-linear workloads and show increased mistakes and decreased reliability. Effective use of resources is yet another essential component of scalable microservices. With the lowest overprovisioning (4%), QGPR attains the highest CPU and network utilisation (79%) and 84%, demonstrating its ability to better scalability and match resource allocation with demand. Due to less efficient resource allocation techniques, Gaussian Process Regression (GPR) and NN lag behind Quantum Support Vector Machine (QSVM), which comes in second. With the lowest SLO violation rate of less than 1% and the quickest adaption time of 10 seconds, QGPR excels in backend system scalability, which also necessitates flexibility in response to workload variations. A crucial benefit in microservices setups, this responsiveness guarantees steady performance even under interference and peak loads. While classic models like GPR and NN show slower adaptation and higher violation rates, making them less appropriate for highly dynamic circumstances, QSVM also shows strong adaptability. For microservices to make decisions in real time, computational efficiency is equally crucial. QGPR strikes the ideal balance for real-time operations by achieving the fastest prediction time of 25 milliseconds while keeping a respectable training time of 28 seconds. The usefulness of NN models for scaled backend systems is undermined by their slower prediction speed and accuracy, despite their quicker training times.

Microservices depend on performance guarantees to remain reliable, and QGPR sets the standard with the lowest tail latency of 210 ms and the greatest SLO compliance rate of 99%. These figures show that it can maintain performance in a variety of scenarios, guaranteeing smooth backend application functioning. While QSVM yields competitive results, GPR and NN models continually fall short of high-performance standards because to their lower SLO compliance and higher tail latency. Overall, the results demonstrate QGPR's remarkable capacity to maximise backend microservices' scalability and performance. For resource scaling in dynamic contexts, QGPR is the most dependable model because it combines accuracy, effectiveness, and flexibility. These findings open the door for more reliable and scalable microservices architectures by demonstrating how quantum-enhanced models may be used to handle the increasing complexity and needs of backend systems.

Conclusion

The study's conclusions highlight how crucial it is to use sophisticated predictive models to improve the scalability and performance of microservices for backend applications. With a Mean Absolute Percentage Error (MAPE) of 5.5% and an R2 score of 0.97, the Quantum Gaussian Process Regression (QGPR) model outperformed more conventional models like Neural Networks (NN) and showed

remarkable technical capabilities among the evaluated approaches. In terms of resource usage, QGPR again performed exceptionally well, with 79% CPU and 84% network utilisation while reducing overprovisioning to just 4%, guaranteeing effective resource matching with changing demands. With the quickest adaption time of 10 seconds and the lowest Service Level Objective (SLO) violation rate of less than 1%, it also demonstrated exceptional flexibility in response to workload variations. QGPR is perfect for real-time decision making in microservices since it matched computational efficiency and operational demands with a prediction time of 25 milliseconds and a manageable training overhead. With the lowest tail latency of 210 milliseconds and a 99% SLO compliance rate, it also offered the most dependable performance assurances. These findings demonstrate QGPR's unparalleled capacity to maximise dependability, performance, and resource allocation in dynamic backend settings, solidifying its position as the go-to paradigm for scaling and overseeing contemporary microservices architectures.

References

1. Karim Eddin S, Salloum H, Shahin MN, Salloum B, Mazzara M, et al. (2024) Quantum Microservices: Transforming Software Architecture with Quantum Computing. Advanced Information Networking and Applications 227-237.
2. M. Villamizar, Oscar Garcés, Lina Ochoa, Harold Castro, Lorena Salamanca, et al. (2016) Infrastructure cost comparison of running web applications in the cloud using AWS lambda and monolithic and microservice architectures. in 2016 16th IEEE/ ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), IEEE 179- 182.
3. Mazzara M, Dragoni N, Bucchiarone A, Giaretta A, Larsen ST, et al. (2018) Microservices: Migration of a mission critical system. IEEE Trans Serv Comput 14: 1464-1477.
4. Lewis J, Fowler M (2014) A definition of this new architectural term.
5. Dragoni N, Saverio Giallorenzo, Alberto Lluch Lafuente, Manuel Mazzara, Fabrizio Montesi, et al. (2017) Microservices: yesterday, today, and tomorrow. Present ulterior Softw Eng 195-216.
6. Dunjko V, Briegel HJ (2018) Machine learning & artificial intelligence in the quantum domain: a review of recent progress. Reports Prog Phys 81: 74001.
7. Biamonte J, Wittek P, Pancotti N, Rebentrost P, Wiebe N, et al. (2017) Quantum machine learning. Nature 549: 195-202.
8. Schuld M (2018) Supervised learning with quantum computers. Springer.
9. Cherukuri BR (2024) Building Scalable Web Applications: Best Practices for Backend Architecture. International Journal of Science and Research 13: 126-139.
10. Aslam N, Sadi Badi (2024) Microservices and DevOps for Fintech: Achieving Cost Efficiency, Agility, and High-Performance Backend Systems.
11. Oyeniran CO, Adewusi AO, Adeleke AG, Akwawa LA, Azubuko CF (2024) Microservices architecture in cloud-native applications: Design patterns and scalability. Comput Sci IT Res J 5: 2107-2124.
12. Mirza D, Muskan Rasool (2024) Balancing User Experience (UX) and Backend Performance in Fintech: Leveraging Cloud-Based Server Management and Microservices.
13. Kaushik N, Kumar H, Raj V (2024) Micro Frontend Based Performance Improvement and Prediction for Microservices Using Machine Learning. J Grid Comput 22: 44.
14. Wasif G, Nadeem Sano (2024) Optimizing Agility and Scalability in Fintech Development: A Microservices Architecture and DevOps Approach.
15. Jerbi S, Gyurik C, Marshall SC, Molteni R, Dunjko V (2024) Shadows of quantum machine learning. Nat Commun 15: 5676, 2024.
16. Gil-Fuster E, Eisert J, Bravo-Prieto C (2024) Understanding quantum machine learning also requires rethinking generalization. Nat Commun 15: 2277.
17. Tüysüz C, Chang SY, Demidik M, Jansen K, Vallecorsa S, et al. (2024) Symmetry Breaking in Geometric Quantum Machine Learning in the Presence of Noise. PRX Quantum 5: 30314.
18. Abbas H (2024) Quantum Machine LearningModels and Algorithms: Studying quantum machine learning models and algorithms for leveraging quantum computing advantages in data analysis, pattern recognition, and optimization. Aust J Mach Learn Res Appl 4: 221-232.
19. Shi J, Shang F, Zhou S, Zhang X, Ping G (2024) Applications of quantum machine learning in large-scale e-commerce recommendation systems: Enhancing efficiency and accuracy. J Ind Eng Appl Sci 2: 90-103.
20. Joseph S, Joshi H, Hassan MM, Bairagi AK (2024) Advancing Quantum Machine Learning: From Theoretical Concepts to Experimental Implementations. Available SSRN 4946682.
21. Kavitha SS, Kaulgud N (2024) Quantum machine learning for support vector machine classification. Evol Intell 17: 819-828.
22. Amirul bin Abdullah, bin Yusuf M (2023) Refining Distributed System Efficiency with Microservices: Advanced Strategies for Enhancing Performance, Scalability, and Resilience in Complex Architectural Environments. J Sustain Technol Infrastruct Plan 7: 16-30.
23. Somashekar G (2023) Proposal: Performance Management of Large-Scale Microservices Applications. Stony Brook University.
24. Ortiz I (2023) Strategic Approaches to Building Highly Scalable, Modular, and FaultTolerant Microservices: Enhancing Application Development, Deployment Efficiency, and Long-Term Maintainability in Modern Distributed Systems. Int J Soc Anal 8: 17-41.
25. Nielsen MA, Chuang IL (2001) Quantum computation and quantum information. Cambridge university press Cambridge 2.
26. Shor PW (1994) Algorithms for quantum computation: discrete logarithms and factoring. in Proceedings 35th annual symposium on foundations of computer science, Ieee 124-134.
27. Ekert A, Jozsa R (1996) Quantum computation and Shor's factoring algorithm. Rev Mod Phys 68: 733.
28. Montanaro A (2016) Quantum algorithms: an overview. npj Quantum Inf 2: 1-8.
29. Dowling JP, Milburn GJ (2003) Quantum technology: the second quantum revolution. Philos Trans R Soc London Ser A Math Phys Eng Sci 361: 1655-1674.
30. Cortes C (1995) Support-Vector Networks. Mach Learn.
31. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521: 436-444.
32. Abdi H, Williams LJ (2010) Principal component analysis. Wiley Interdiscip Rev Comput Stat 2: 433-459.
33. Breiman L (2001) Random forests. Mach Learn 45: 5-32.
34. Cover T, Hart P (1967) Nearest neighbor pattern classification. IEEE Trans Inf theory 13: 21-27.
35. Hosmer Jr DW, Lemeshow S, Sturdivant RX (2013) Applied logistic regression. John Wiley & Sons.
36. Huang L, Qin J, Zhou Y, Zhu F, Liu L, Shao L (2023) Normalization Techniques in Training DNNs: Methodology, Analysis and Application. IEEE Trans Pattern Anal Mach Intell 45: 10173-10196.
37. Begum BA, Nandury SV (2023) Data aggregation protocols for WSN and IoT applications - A comprehensive survey. J King

Saud Univ- Comput Inf Sci 35: 651-681.

38. Cembaluk P, Aniszewski J (2022) Forecasting the network traffic with PROPHET. 3rd Polish Conf Artif Intell 215-218.
39. Hussain B, Afzal MK, Ahmad S, Mostafa AM (2021) Intelligent traffic flow prediction using optimized GRU model. IEEE Access 9: 100736-100746.