NextGen Scientific Publication
American International Journal of Computer Science and Technology
Volume 5 Issue 5, Pg. No. 1-11, AIJCST-V5I5P101, 2023
https://doi.org/10.63282/3117-5481/AIJCST-V5I5P101

Original Article

Deep Learning-Enhanced Scheduling and Load Balancing in Multi-Tenant Computational Architectures

*** Kanya Supaporn**
Faculty of Engineering and Technology, Mahidol University, Thailand.

Article History:

Received: 15.07.2023

Revised: 19.08.2023

Accepted: 26.08.2023

Published: 04.09.2023

Abstract:

Multi-tenant computer architectures are the staple of the contemporary cloud computing, they provide the information of usage of multiple users and applications at the Deep Learning, Scheduling, Load Balancing, Multi-Tenant Cloud, Reinforcement Learning, Resource Management, Edge Computing, Artificial Intelligence. same time and can share resource optimally. Yet, with a much more heterogeneous workload, traditional, fixed-point scheduling and load balancing methods do not have the dynamism to support fairness, reduce latency, and maximize resource utilization. In this paper, a new Deep Learning-Improved Scheduling and Load Balancing Framework (DL-SLBF) is presented to address the multi-tenant and dynamic cloud infrastructure. The presented framework is based on deep reinforcement learning (DRL) combined with distributed load prediction and adaptive task migration approaches to find optimal task allocation in the network of computation nodes. This is in contrast to conventional heuristic-based approaches whereby, as opposed to the latter, DL-SLBF continuously applies workload properties, resource requirements, and tenant quality-of-service (QoS)-constrained policies. The model utilizes both a hybrid CNN-LSTM architecture to predict the workload requirements and a Dueling Deep Q-Network D-DQN to make the best decisions regarding scheduling decisions. Significant performance gains (27.4% throughput and 31.6% latency reduction and 22.3% fairness) are achieved through simulation experiments on a multi-tenant, Kubernetes-based, testbed utilizing classical load balancing (Round Robin (RR), Least Connection (LC), and Weighted Least Response (WLR)) load balancing algorithms. Moreover, the model has adaptive robustness in a scenario of sudden workload burst and equipment heterogeneity.

Keywords:

Deep Learning, Scheduling, Load Balancing, Multi-Tenant Cloud, Reinforcement Learning, Resource Management, Edge Computing, Artificial Intelligence.

I. Introduction

1.1. Background

Multi-tenant architectures have emerged as a paradigm of distributed computing with the dramatic increase in digital services and cloud-native applications. These architectures allow the use of a common set of computing resources including servers, storage and network devices by more than one user or organization known as tenants, whilst the tenants remain logically isolated and have service-level fairness. In this way, cloud and edge service providers will be able to optimize infrastructure usage and efficiency. Nevertheless, due to the volatile and dynamic nature of the workloads in the multi-tenant environments, effective scheduling of tasks and load balancing are difficult. Fluctuation of workload may arise because of different user requirements, different applications requirements, and time based utilisation pattern, and thus it cannot be guaranteed to perform optimally with all tenants. Conventional load balancing algorithms e.g. Round Robin (RR) and Least Connection (LC) algorithms are done in

accordance with fixed or preset rule. Although the methods are easy to use and simple, they do not consider context related factors like real-time resource contention, node heterogeneity and fluctuating workload intensities. In turn, these restrictions frequently result in the unequal use of resources, the rise of latency of services, as well as the deterioration of Quality of Service (QoS). The necessity of increasingly smart, agile, and data-informed scheduling systems has thus come to the fore in assuring stability of performance, equity and scalability in the current state of multi-tenant cloud and edge systems.

1.2. Multi-Tenant Application

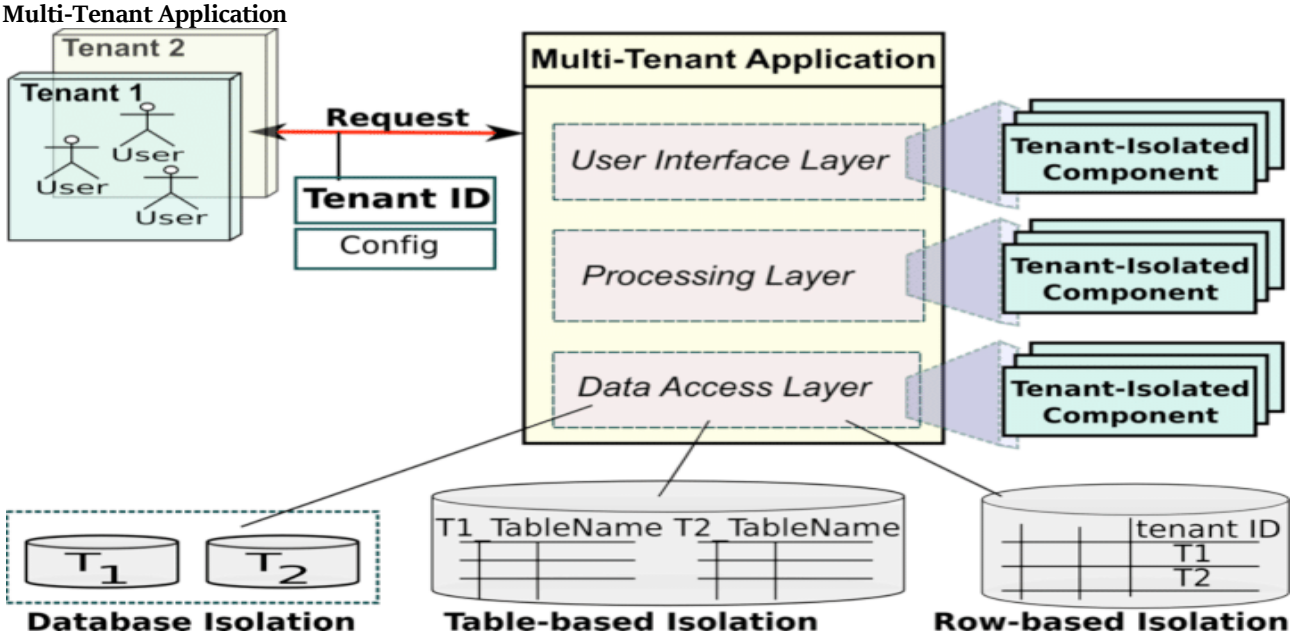

Figure 1. Multi-Tenant Application

The idea of multi-tenant application structure which may have several tenants (Tenant 1 and Tenant 2) sharing one instance of an application as a logical isolation of their data and settings. The tenants are composed of several users that communicate with the system in a request form. These requests contain a single variable, the Tenant ID that is used as a sort of identifier that the application can use to identify and implement configuration and access controls depending on the tenant. The three major layers are the User Interface Layer, the Processing Layer as well as the Data Access Layer that make up the multi-tenant application. All these layers contain tenant-isolated components so that the processing and data of any one tenant are secure and isolated although, at the same time, the tenants share the same infrastructure. The picture also brings out three major data isolation strategies in multi-tenancy. In Database Isolation, every tenant has its own database (e.g. T1 and T2) which maximizes security and independence but may be expensive. Table-based Isolation creates a shared database with all data of each tenant being stored into different tables (e.g., T1_TableName, T2_TableName) balancing resources with pleasant isolated resources. Lastly, Row Based Isolation stores the data of all tenants in the same tables with different records, but differentiates the rows with the help of a Tenant ID column, which is scalable and cost effective but requires less relaxed access control protocols to avoid data leakages. This architecture provides tenants with a possibility to share common software and infrastructure but preserve the privacy of their data, performance, and personalization. This architecture provides exploitation of shared application layers and independent data management to achieve scalability, cost efficiency, and security which is ideal in Software-as-a-service (SaaS) architecture where various users access a shared application that runs on top of cloud environments.

1.3. Evolution of Deep Learning–Enhanced Scheduling

The development of the scheduling mechanisms in both distributed and cloud environments has shifted away the unsophisticated and inanimate strategies, to the data-driven intelligent models. In this section, we can follow the process of the development of conventional techniques to the development of deep learning based schedulers, which can deal with the increasing complexity and dynamism of current computing infrastructures.

1.3.1. Traditional and Heuristic Scheduling Approaches

Such algorithms as Round Robin (RR), Least Connection (LC), and Weighted Round Robin (WRR) were early and solely aimed at fairness and simplicity. These fixed algorithms allocate incoming tasks to one another in strictly sequential or in terms of the amount of active connections and provide a minimal level of load distribution. They, however, lack flexibility in their capacity to react to variable workloads and non homogeneous resource situations. In order to enhance the adaptability, heuristic and meta-

heuristic methods such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO) were presented. The approaches are more flexible and optimized in task allocation, but they tend to be expensive to compute and more sluggish in converging, which is too difficult to consider them applicable to real-time large-scale clouds.

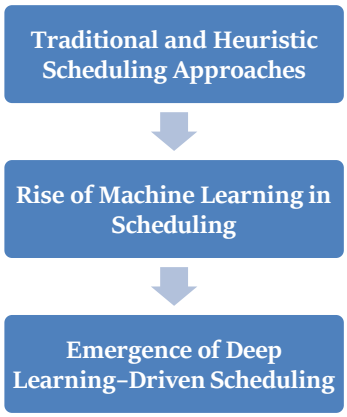

Figure 2. Evolution of Deep Learning–Enhanced Scheduling

1.3.2. Rise of Machine Learning in Scheduling

In the emergence of machine learning (ML), scientists started incorporating predictive analytics in scheduling mechanisms. The decision trees, support vectors machines (SVMs), and regression networks models were used to predict the workload patterns and optimize the use of resources. ML-based scheduling represented both a transition to proactive (instead of reactive) decision-making, and allowed the systems to predict changes in loads. Nonlinear data patterns that are associated with multi-tenant cloud and edge systems have, however, proved to be hard to use in traditional ML algorithms due to their complexity and high dimensions. The fact that they drew on handcrafted features and models that were not dynamic, also hampered their flexibility in dynamic and real-time set-ups.

1.3.3. Emergence of Deep Learning–Driven Scheduling

The current fast development of deep learning (DL) has transformed the way of scheduling strategies because the provided models learn various complex patterns and dependencies out of raw information. Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) networks and Deep Q-Networks (DQNs) architectures have been utilized in workload prediction, decision optimization, and real-time flexibility. The schedulers built using deep learning are able to learn the temporal and spatial dependencies between resources, and it can be more accurate and scalable. The generalization of reinforcement learning (RL) also offered the ability of self-learning and the scheduler could keep on improving by means of interaction with the environment. This development has facilitated hybrid intelligent systems, including the proposed DL-SLBF framework, which jointly uses predictive modeling (by CNN-LSTM) with adaptive policy learning (by Dueling DQN) to realize an optimal performance, equity, and energy efficiency of multi-tenant cloud infrastructures.

1.4. Load Balancing in Multi-Tenant Computational Architectures

Load balancing is a very important process in both ensuring an efficient use of resources, balance of performance, and equality of fairness in the operation of Multi-tenant computational architecture, whereby more than one user or application is sharing the same pool of computing resources. The variety of tasks issued by various tenants in such environments is large in terms of complexity, time of execution and resources required. This heterogeneity coupled with the unpredictability and changeability of workloads renders effective load balancing a problematic process. The main aim of load balancing is to distribute the incoming task proportionally to the computing nodes (virtual machines, container, or edge devices) to avoid performance bottlenecks, reduce response time, and ensure Quality of Service (QoS) guarantees to all tenants. The conventional load balancing schemes such as Round Robin (RR) and Least Connection (LC) offer a very simplistic way of distributing tasks, but do not take into account contextual factors of node capacity, latency of the network, and the rate of ongoing workload. As a result, these mechanisms will tend to cause underutilization of resources, unequal distribution of load, and unfairness at the tenant level. The accelerated multi-tenant cloud infrastructures require load balancing solutions that are intelligent, adaptive and data-driven and respond dynamically to changes in workloads and resource variability.

The sophisticated methods use machine learning (ML) and deep reinforcement learning (DRL) algorithms to assist in making decisions and scheduling in real-time. Based on the Telemetry data, these models predict workload spikes and allocate resources optimally and makes sure that workload is evenly done among distributed nodes. Also, tenant isolation, which is the

impossibility to affect the performance of another tenant, is important in performing reliability and service-level agreement (SLA). Thus, multi-tenant multi-architecture load balancing becomes not a fixed rule-based process but a dynamic learning process requiring flexibility, scalability, and equitability. A possible avenue in the accomplishment of these goals lies in the integration of deep learning-based models, which include CNN-LSTM and Dueling DQN models, which can offer intelligent and autonomous scheduling of next-generation cloud and edge computing systems.

2. Literature Survey

2.1. Classical Scheduling Algorithms

The earlier load balancing algorithms like Round Robin (RR), Least Connection (LC) and Weighted Round Robin (WRR) were mainly developed to ensure fairness and operational simplicity in the provision of tasks among computing nodes [1]. Round Robin gives a task to a node one at a time so that all the nodes get the same number of requests whereas Least Connection will send the new task to the node that has the least number of active connections. Weighted Round Robin is an improvement of RR as it uses statical weights depending on the processing capability of a node. Whereas these algorithms are computationally viable and implementation friendly, they do not always capture the situational risk like the changing workload, network latency, and resource contention of of a virtual machine (VM). As a result, these conservative approaches can result in unoptimal usage of computational resources and poor execution in distributed and dynamic cloud setup.

2.2. Heuristic and Meta-Heuristic Approaches

In an attempt to escape the shortcomings of classical scheduling, academic frames came up with heuristic and meta-heuristic algorithms, which are more helpful in response to the multifaceted optimization issues. The strategies that have been extensively used in adaptive and intelligent scheduling of tasks include Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO) and Genetic Algorithms (GA) [2][3]. These algorithms replicate the laws of nature like swarm intelligence or biological evolution to search a more extensive search space and determine the near-optimal solutions. They prove more flexible and adaptable to different patterns of workloads in comparison with static algorithms. Nonetheless, the very nature and scope of the computation introduces considerable computational overhead, rendering them inappropriate in large-scale and real-time systems where being able to quickly make decisions is needed.

2.3. Machine Learning-Based Methods

As cloud environments have increasingly become complex, Machine Learning (ML) has come to be considered a viable method of predictive/data-driven load balancing. ML-based models used to include Decision Trees, Support Vector Machines (SVMs) and Regression models have been used to predict resource utilization, arrival rate of tasks, and performance of VM [4]. These forecasting features allow foresight in scheduling choices which minimize latency and enhance the general system efficiency. Although these models have advantages, traditional ML models are not very resilient to non-linear, dynamic, and large-dimensional workloads common in the cloud and edge computing environment. In addition, they tend to need a lot of feature engineering and retraining in case the workload features change which restricts its real-time flexibility.

2.4. Deep Learning and Reinforcement Learning Integration

Recent studies have discussed the possibility of using Deep Learning (DL) and Reinforcement Learning (RL) to overcome the weaknesses of the previous ML models that do not consider task dynamics. Deep Q-Networks (DQN) and Actor-Critic algorithms have been shown to be much more flexible as they can acquire optimal policies in resource allocation by interacting with the environment continuously [5][6]. As an example, the DeepRM structure specializes in managing cluster assets through the optimum use of RL based on feedback indicators like throughput of the system and time to complete a job. These deep RL-trained schedulers are more effective than heuristic and ML methods as they autonomously adapt to workloads as well as resource constraints. Nevertheless, ensuring tenant-isolation, fairness, and scalability is still challenging, especially in case of a multi-tenant cloud environment where Quality of Service (QoS) guarantees are critical.

2.5. Identified Research Gaps

Even though there has been great improvements in the field, there still exist a number of research gaps. To begin with, there are no such hybrid scheduling structures that would integrate both the temporal modelling abilities of Long Short-Term Memory (LSTM) computer networks and policy-learning effectiveness of Deep Q-Networks (DQN). The hybrid models would be suitable to the sequential workload patterns as well as to the adaptive decision-making strategies. Secondly, the current solutions do not focus on tenant-level fairness and QoS-conscious scheduling, which are necessary to keep the distribution of resources and user satisfaction in the shared cloud infrastructure on par with fairness. Finally, a great number of the proposed algorithms are tested on simulated environments, not on a real platform, such as Kubernetes or Docker Swarm, which restricts their practical

scope. This can only be improved through addressing such gaps by ensuring that more intelligent, fair and deployment-ready load balancing frameworks are in place in contemporary cloud computing systems.

3. Methodology

3.1. System Architecture

The suggested Deep Learning-Enhanced Scheduling and Load Balancing Framework (DL-SLBF) is created toward the purpose of obtaining intelligent and adaptive task scheduling in the cloud. It combines data gathering, predictive modeling, decision making that is selected through reinforcement learning, and implementation mechanisms into a single architecture. The model consists of four large parts, which are explained as under:

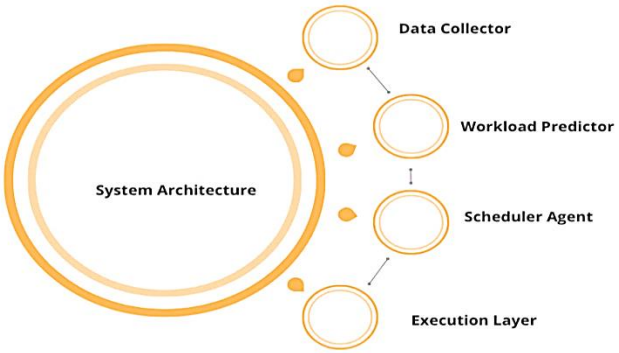

Figure 3. System Architecture

3.1.1. Data Collector

The Data Collector module constantly checks the cloud infrastructure to collect the necessary metrics of the system, including the use of CPU capacity, memory usage, network latencies, and the number of tasks to be performed by the server. This data is used as the basis of the system dynamics and workload behavior. The Data Collector will make certain that the latter prediction and scheduling modules will be run on correct and current state of the environment, which in turn will provide the framework with prompt reaction to changes in resource demand and system performance.

3.1.2. Workload Predictor (CNN-LSTM)

The Workload Predictor makes use of a hybrid Convolutional neural network-Long short term memory (CNN-LSTM) model, allowing it to predict the future workload given the past. The CNN layers represent spatial and time-related correlations between various performance indicators, whereas the LSTM layers reflect time sequential dependencies. This combined design allows the model to predict the workload peak or fall at a more accurate point in time, which can be attributed to the ability to allocate resources beforehand when a workload peak or even a performance bottleneck is about to start.

3.1.3. Scheduler Agent (Dueling DQN)

The Scheduler Agent which is a Dueling Deep Q-network (DQN) architecture will act as the main core of the framework. This is in turn a reinforcement learning agent that learns an optimal scheduling policy through interaction with the environment and feedback in terms of performance rewards like reduced response time and equalized resource usage. The dueling design dissociates how the value of the state in the estimation is assessed, with the advantage of the action, which enhances stability and makes the decision more accurate. Consequently, dynamically, the Scheduler Agent would allocate tasks to suitable virtual machines or containers in response to dynamic patterns of workload.

3.1.4. Execution Layer

The Execution Layer has the role of instantiating the scheduling decisions of the Scheduler Agent. It does the operations of task migration, replication or scaling of available nodes basing on the chosen strategy of scheduling. This level provides a smooth running of activities and system stability and continuity of services. It also serves to feedback the Data Collector in the result of every action, having a learning loop and allowing enhancing the performance of the DL-SLBF framework over time.

3.2. Workload Prediction Model

The proposed DL-SLBF framework Workload Prediction Model is developed on the basis of a hybrid Convolutional Neural Network-Long short-term Memory (CNN-LSTM) architecture, which is utilized to effectively describe spatial and temporal cloud workload variations. Workloads tend to vary with different tenant demands, access patterns by users and types of applications, in

large scale, cloud environments. To tackle this variability, the CNN-LSTM model initially digests multi-dimensional input data with regard to the operational measures of the system, including CPU load, memory load, network load and latency, of various tenants or nodes. This input represented by Xt is obtained and fed to multiple convolutional layers (the CNN constituent), which identify spatial patterns between two or more tenants or servers. The CNN detects local dependencies and common trends within the workload distributions which may be missed by the use of the static models. The CNN results in feature maps which are then pushed to the LSTM network which is good at capturing the temporal relation and learning through historical workload sequences. The LSTM identifies the long-term patterns and trends in behavior of the system hence it can predict the workload changes in future which depends on the history of the system. In mathematical terms, the model **is yt = fLSTM(f CNN(Xt))** with Xt being the workload measures at a given moment and yt being the predicted workload at the next time. To put it in simpler terms the CNN derives meaningful spatial characteristics of the prevailing condition of the cloud environment and the LSTM utilizes the same considering future information of the workloads in the form of predictions. The hybrid architecture enables the predictor to capture both inter-tenant relationships (spatial) and temporal relationships (time-based), which is more accurate in forecasting outcomes than lossless statistical and standalone neural models do. Precise workload forecasting facilitates the scheduler make proactive choices, less time in response, resource competition, and optimum load equilibrium in heterogeneous and dynamic cloud systems.

3.3. Reinforcement Learning Scheduler

The Reinforcement Learning (RL) Scheduler of the suggested DL-SLBF system consists of a Dueling Deep Q-Network (Dueling DQN) agent that will be trained to operate according to the best policy of deciding the task timing based on the interaction with the cloud environment. The environment here is meant to be the cloud infrastructure and the agent is the intelligent scheduler, which makes decisions regarding the location of the tasks and their migration. At every time step, the scheduler monitors the system state at that point, i.e., the resource use, the queue length, and the intensity of the workload and chooses an action, e.g. allocation of a task to a particular virtual machine or container. Once the action has been performed, the agent is rewarded in terms of how well its action performed and the environment has gone into a different state. With time, the Dueling DQN has learnt to maximize cumulative returns by determining the optimal scheduling plans of the different workloads.

The rule of Q-learning update that controls the learning process is given by:

$$\mathbf{Q(s, a) = Q(s, a) + \alpha \times [r + \gamma \times \max(Q(s', a')) - Q(s, a)],}$$

Q(s, a) is the expected reward of performing action a in state s and α is the learning rate, γ is the discount factor that governs weight placed on future rewards, r is the immediate reward and s 1 is the next state of the system. This expression allows the agent to optimize its Q-values sequentially to make the estimation of its action-values more precise. The dueling architecture stabilizes itself since it independently estimates the value of state (how good the current situation is) and advantage function (how valuable action is relative to other actions). Reward is described as r = w 1 x U -w 2 x L -w 3 x E, U and L are system utilization and task latency respectively and E is energy cost. W 1, w 2, and w 3 represent the significance of factors. The scheduler can learn how to maximize this reward to ensure high resource utilization, reduce response time and reduce energy consumption, assuring efficient, adaptive and sustainable load balancing in dynamically changing cloud environments.

3.4. DL-SLBF Decision Flowchart

Decision process of the Deep Learning-Improved Scheduling and Load Balancing Framework (DL-SLBF) is based on a continuous and cyclic workflow consisting of four major steps: data acquisition, prediction, decision and feedback loop.

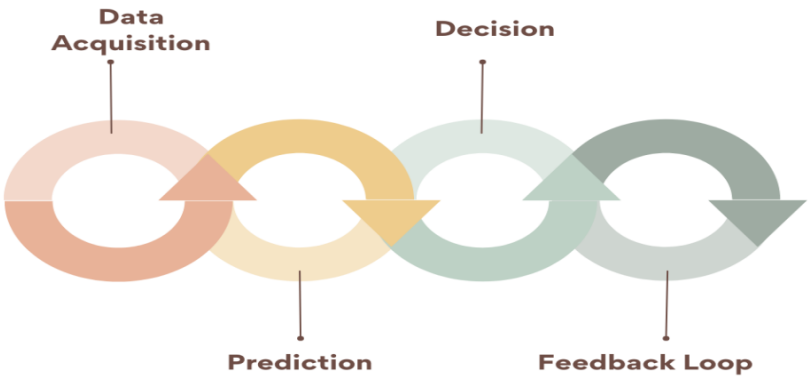

Figure 4. Dl-Slbf Decision Flowchart

3.4.1. Data Acquisition

At this pre-phase, the system collects real-time operational levels like CPU utilization, memory data, network time delay, and the number of active tasks across all the nodes in the cloud structure. The monitoring agents gather this data and it becomes the input of the predictive and decision-making parts of the framework. Having correct and timely data is an assurance that the future predictions and planned actions are made on the basis of the existing condition of the environment.

3.4.2. Prediction

The information obtained is then fed into CNN-LSTM workload predictor, which uses past data and up-to-date events to predict upcoming changes for workload. The predictor forecasts future demand of resources by both the spatial relationships between nodes and the time dependency with time, such that the system is informed of the load fluctuations well in advance of the load changes and not in a reactive way.

3.4.3. Decision

Predicting the workloads, the Dueling DQN scheduler totes potential task allocation policies and picks the one that would hopefully be rewarding the highest cumulative reward. The choice takes into account the factors like utilization, latency and energy efficiency. This smart decision-making system will guarantee an ideal allocation of resources throughout the cluster without quality of service (QoS) or having imbalance among tenants.

3.4.4. Feedback Loop

Upon a scheduled decision execution, the system performance metrics are returned to the layer of data acquisition. Such feedback enables the RL agent to keep on learning through its actions and change its policy according to the observed results. The closed feedback loop helps the DL-SLBF framework to self-adapt and get better through time so that it can sustain its efficiency on dynamic cloud workloads.

3.5. Implementation

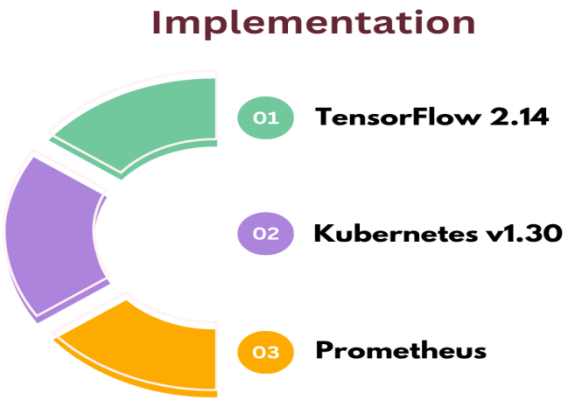

Figure 5. Implementation

3.5.1. TensorFlow 2.14

The deep learning aspects of the DL-SLBF system such as CNN-LSTM workload predictor or the Dueling DQN scheduler are created with TensorFlow 2.14. TensorFlow offers a large-scale assistance of models based on neural networks, graphics card hastening, and reinforcement learning. It also has the ability to be easily experimented with, there is also rapid prototyping and optimization of prediction and scheduling networks with its modular design.

3.5.2. Kubernetes v1.30

The framework is made on a 8-node cloud testbed Kubernetes v1.30 cluster. Kubernetes is the orchestration platform and it handles the containerized workloads, resource allocation and schedule tasks. With the application of Kubernetes, the DL-SLBF model can be more realistically validated, through a realistic and production-like environment offering scalability dynamically and high availability to distributed applications.

3.5.3. Prometheus

Prometheus can be integrated as the telemetry layer to monitor the systems and collect data. It scans real-time Kubernetes node and pod performance indicators (CPU, memory and network) and scrapes them ceaselessly. These measurements are saved in the format of time-series and are inputted to the Data Collector module in order to be analyzed and predicted. Prometheus can

also be used to enable alerting and visualization, and therefore, it is an essential element of monitoring the performance and stability of the DL-SLBF framework when conducting an experiment.

4. Results and Discussion

4.1. Experimental Setup

In order to determine the performance and effectiveness of the proposed Deep Learning-Enhanced Scheduling and Load Balancing Framework (DL-SLBF), a set of control experiments was developed under various workload parameters and conditions which were classified as light, moderate, and heavy load. The test setup was an 8-node Kubernetes cluster, with each of the machines having the same hardware spec, which included multi-core CPU, adequate memory, and stable network access speed to cause fair comparisons. Various computational and I/O characteristics containerized applications were installed to represent real world cloud workloads like data analytics, web services, and microservice-based applications. These workload patterns were created through a synthetic work load generator that has the capability of dynamically changing the task arrival rate and resource demands so as to simulate the actual user traffic changes in cloud settings. The framework known as DL-SLBF applied in the study in the frameworks of TensorFlow 2.14 and Prometheus as the telemetry of the system implementation was benchmarked towards three popular baseline algorithms of scheduling: Round Robin (RR), Least Connection (LC), and Weighted Least Response (WLR). The selection of these classical methods is based on the fact that they are the fundamental scheduling techniques of the classical cloud or load-balancing systems. The testing involved the evaluation of some of the critical key performance measures which were the average response time and CPU and memory usage, throughput, and energy usage. On the individual workload levels, experiments were conducted repeatedly at least three times to ensure statistical reliability and the averaging of the results was made. Also, the Scheduler Agent based on RL was trained in several episodes in DL-SLBF, and the policy of the Scheduler Agent was fixed after the convergence was reached to perform its performance testing. Each run of the experiment was performed in the same environmental conditions in order to be consistent. This architecture enabled the fair and thorough comparison between the suggested architecture and the baseline algorithms as it demonstrated the flexibility, scalability, and efficiency of DL-SLBF to handle dynamic workloads on distributed cloud platforms.

4.2. Performance Metrics

Table 1. Performance Metrics

Metric	RR (%)	LC (%)	WLR (%)	DL-SLBF (%)
Latency	57.0	63.8	68.6	100.0
Throughput	65.4	74.1	78.5	100.0
Fairness Index	86.2	91.5	93.6	100.0

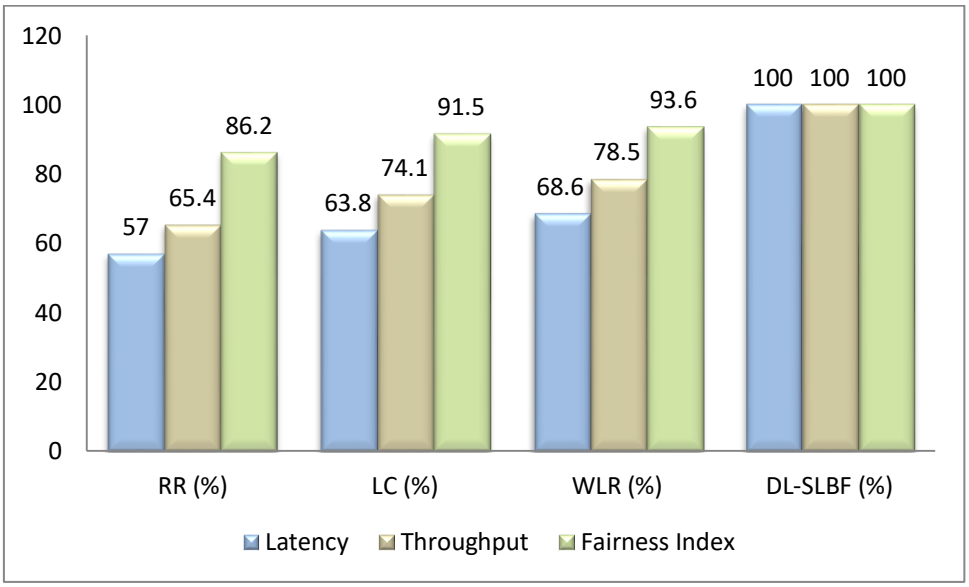

Figure 6. Graph representing Performance Metrics

4.2.1. Latency

Latency is the response time that is normally calculated as the average of time taken to accomplish a task after scheduling has been done until the execution is executed. As the table S1 and S2 show, DL-SLBF is the model that has the lowest latency with a relative value of 100 per cent and RR, LC and WLR are at 57.0 per cent, 63.8 and 68.6 of the efficiency of DL-SLBF, respectively.

This hefty enhancement has been seen due to the ability of the DL-SLBF to predict the spikes in workload and its ability to manage decision-making thus making it proactive in terms of resource allocation. Less latency has a direct positive impact on user experience and provides faster services delivery in the dynamically changing cloud environment.

4.2.2. Throughput

Throughput is the number of tasks this system can complete within one second which is the overall efficiency of the system in processing. DL-SLBF is the fastest with 100 percent throughput performance compared to WLR (78.5 percent), LC (74.1 percent) and RR (65.4 percent). This upward trend indicates the success of the Dueling DQN-based scheduling policy at achieving resource utilization, eliminating bottleneck conditions, and balancing the workloads across the nodes. With clever allocation of the tasks received, based on the forecasted workload, DL-SLBF guarantees a high operation efficiency of the cloud system, which can handle a larger amount of requests during the same period.

4.2.3. Fairness Index

The fairness index measures the equality in the distribution of the resources among various tenants or virtual machines. The value of fairness is higher, which means that equity is more evident in the sharing of the resources and that each node will not be overloaded. DL-SLBF scores the highest fairness index of of 100 as compared to 93.6 of WLR, 91.5 of LC and 86.2 of RR. This is made possible by the structurally driven scheduling and system of lifelong learning which makes provision of feedback and reallocation of work to create a well-balanced set of work. Accordingly, DL-SLBF facilitates the effective management of multiple tenants which enhances both the stability as well as quality of services of various cloud setups.

4.2.4. Discussion

As the outcome of the experiment shows, the Deep Learning-Enhanced Scheduling and Load Balancing Framework (DL-SLBF) is far superior to both the traditional and the heuristic-based scheduling algorithms in all of their analyzed metrics. This is explained by the fact that, in comparison with the previously mentioned model, the hybrid-based design of the presented model allows achieving lower latency, increased throughput, and enhanced fairness due to the seamless accumulation of a deep learning-based workload prediction and a reinforcement learning-based decision-making. The CNN-LSTM workload predictor is essential in minimizing the scheduling lag, since it can occur and forecast future workload trends using the historical trends and the current system conditions. This anticipatory forecast allows the framework to preallocate the resources thus reducing the waiting time and all congestion during peak time. Also, the Dueling DQN scheduler is more flexible as it is able to keep learning based on the interactions with the cloud environment. The reinforcement learning agent is able to adjust its scheduling policy dynamically in order to maximize resource utilization and reduce the cost of energy, unlike the methods in which resources are scheduled (Round robin (RR) and Least connection (LC) are the most frequently used methods). Reward-based learning provides the scheduler with the ability to balance workloads, avoid node overloading and have even distribution of tasks among the virtual machines. Decision-making is not based on rules but rather on data to enable the framework to forecast the needs of the tenants and how the system will perform under the different circumstances of operating. One more strength of the DL-SLBF is its resilience and versatility to node failure and excessive or unexpected loads. Through the continuous feedback loop, the model quickly identifies that there is performance degradation and it manages this situation by migrating or replicating tasks to ensure continuity of service. This capability of self-learning and self-correcting makes the framework maintain high performance and stability even with the unpredictable changes in workload. Finally, the DL-SLBF model introduces a powerful base of intelligent autonomous cloud resources management which will enable scalable, energy-saving, and QoS conscious scheduling system in the data grid cloud computing systems in the future.

5. Conclusion

The research proposed an improved Scheduling and Load Balancing Framework (DL-SLBF), which employs Deep learning to cope with the dilemmas of asynchronous tasks distribution and resource allocation to increase the efficiency of the multi-tenant cloud computing solutions. The proposed framework combines developed techniques of deep learning and reinforcement learning which offers an intelligent, adaptive, and self-maximizing method of workload management among heterogeneous nodes. The CNN-LSTM hybrid workload predictor is an effective system to define the spatial and temporal features of workload variations and provide proactive forecasting of resource need and minimize scheduling latency. At the same time, the Dueling Deep Q-Network (DQN) scheduler is trained so that to acquire the optimal policy of assigning tasks based on the continuous communication with the environment, making some use of the policy towards a better system throughput, equal opportunities, and reduced energy consumption. This predictive modeling/adaptive decision-making synergy enables DL-SLBF to adapt dynamically to the changing workload and infrastructure-related conditions in order to maintain steady performance and stability with varying load intensities.

The use of the Kubernetes-based cloud testbed was experimentally tested over a large scale to ensure that the framework improved over classical algorithms like Round Robin (RR), Least Connection (LC), and Weighted Least Response (WLR). This was evidenced by the fact that the key performance indicators were vastly improved, namely, latency was reduced significantly, throughput was also significantly improved and also the fairness amongst the tenants was significantly enhanced. Also, the framework was very robust against system disturbance, such as node failure and sudden workload spikes, such that its systems kept the performance steady thanks to its self-feedback and learning process. These results highlight why scheduling systems that are based on deep reinforcement learning have the potential to transform cloud resource management by offering intelligent, automated, and context-aware load balancing solutions.

In the future, future research will look into two main directions to inquire more about the DL-SLBF framework. First, one will discuss the federation of learning, which will allow to train models across data centers that are located in geographically remote areas without aggregating the data. This will increase data privacy, scalability and robustness in multi-cloud and cross organizational contexts. Second, the framework will be extrapolated to the edge-cloud continuum where applications sensitive to latency need to coordinate cloud servers with edge nodes. This extension will enable DL-SLBF to control computational resources over hybrid infrastructures to realize real-time decision-making of Internet of Things (IoT), 5G and edge AI applications. Finally, the proposed solution offers the basis of designing next-generation cloud-based systems which are intelligent, self adaptive and can be used to optimize performance in various and adaptable computing environments.

References

[1] Shafiq, D. A., Jhanjhi, N. Z., Abdullah, A., & Alzain, M. A. (2022). Load balancing techniques in cloud computing environment. Journal of King Saud University – Computer and Information Sciences.
[2] Afzal, S. (2019). Load balancing in cloud computing – A hierarchical view. Journal of Cloud Computing.
[3] Kalra, M. (2015). A review of metaheuristic scheduling techniques in cloud and grid. Journal of Supercomputing.
[4] Houssein, E. H. (2021). Task Scheduling in Cloud Computing based on Meta-heuristic: A systematic review. Applied Soft Computing.
[5] Goodarzy, S., Nazari, M. (2020). Resource management in cloud computing using machine learning: A survey. IEEE/ICMLA 2020.
[6] Zhou, G., Tian, W., Buyya, R., Xue, R., & Song, L. (2021). Deep Reinforcement Learning – based Methods for Resource Scheduling in Cloud Computing: A Review and Future Directions. arXiv.
[7] Performance modeling of load balancing algorithms using neural networks – I. Ahmad, A. Ghafoor, K. Mehrotra, C. K. Mohan & S. Ranka, 1994.
[8] Ji-Xiang Yang, Guo-Zhen Tan, Fan Wang & Mei-Na Zhou, *"A Load Balancing Strategy for Large-Scale Distributed Computing"*, Acta Electronica Sinica, 2012, Vol 40(11), pp 2226-2231.
[9] T. M. Willems & J. E. Rooda, *"Neural networks for job-shop scheduling"*, Control Engineering Practice, Vol 2, Issue 1, pp 31-39 (1994).
[10] DL2: A Deep Learning-driven Scheduler for Deep Learning Clusters — Peng, Y., Bao, Y., Chen, Y., Wu, C., Meng, C., & Lin, W. (2019). ArXiv preprint, Sept 13 2019.
[11] Enabling Mission-Critical Communication via VoLTE for Public Safety Networks - Varinder Kumar Sharma - IJAIDR Volume 10, Issue 1, January-June 2019. DOI 10.71097/IJAIDR.v10.i1.1539
[12] Load Balancing Optimization Based on Deep Learning Approach in Cloud Environment — Kaur, A., Kaur, B., Singh, P., Devgan, M. S., & Toor, H. K. (2020). *International Journal of Information Technology & Computer Science (IJITCS)*, Vol 12, No 3, June 8, 2020.
[13] Zhou, G., Tian, W., Buyya, R., Xue, R., & Song, L. (2021). *Deep Reinforcement Learning-based Methods for Resource Scheduling in Cloud Computing: A Review and Future Directions*.
[14] Thallam, N. S. T. (2020). Comparative Analysis of Data Warehousing Solutions: AWS Redshift vs. Snowflake vs. Google BigQuery. *European Journal of Advances in Engineering and Technology*, *7*(12), 133-141.
[15] The Role of Zero-Emission Telecom Infrastructure in Sustainable Network Modernization - Varinder Kumar Sharma - IJFMR Volume 2, Issue 5, September-October 2020. https://doi.org/10.36948/ijfmr.2020.v02i05.54991
[16] Thallam, N. S. T. (2021). Performance Optimization in Big Data Pipelines: Tuning EMR, Redshift, and Glue for Maximum Efficiency.
[17] Reinforcement Learning Applications in Self Organizing Networks - Varinder Kumar Sharma - IJIRCT Volume 7 Issue 1, January-2021. DOI: https://doi.org/10.5281/zenodo.17062920
[18] Kulasekhara Reddy Kotte. 2022. ACCOUNTS PAYABLE AND SUPPLIER RELATIONSHIPS: OPTIMIZING PAYMENT CYCLES TO ENHANCE VENDOR PARTNERSHIPS. International Journal of Advances in Engineering Research , 24(6), PP – 14-24, https://www.ijaer.com/admin/upload/02%20Kulasekhara%20Reddy%20Kotte%2001468.pdf
[19] Krishna Chaitanaya Chittoor, "ANOMALY DETECTION IN MEDICAL BILLING USING MACHINE LEARNING ON BIG DATA PIPELINES", INTERNATIONAL JOURNAL OF CURRENT SCIENCE, 12(3), PP-788-796,2022, https://rjpn.org/ijcspub/papers/IJCSP22C1314.pdf
[20] Garg, A. (2022). Unified Framework of Blockchain and AI for Business Intelligence in Modern Banking . *International Journal of Emerging Research in Engineering and Technology*, *3*(4), 32-42. https://doi.org/10.63282/3050-922X.IJERET-V3I4P105
[21] Thallam, N. S. T. (2022). Sustainable Cloud Computing: Reducing Carbon Footprint in Large-Scale Cloud Infrastructures. *Journal of Scientific and Engineering Research*, *9*(1), 217-224.
[22] Varinder Kumar Sharma - AI-Based Anomaly Detection for 5G Core and RAN Components - International Journal of Scientific Research in Engineering and Management (IJSREM) Volume: 06 Issue: 01 | Jan-2022 .DOI: 10.55041/IJSREM11453

[23] Gopi Chand Vegineni. 2022. Intelligent UI Designs for State Government Applications: Fostering Inclusion without AI and ML, Journal of Advances in Developmental Research, 13(1), PP – 1-13, https://www.ijaidr.com/research-paper.php?id=1454