Copyright ⓒ 2023 KSII

Long-Term Container Allocation via Optimized Task Scheduling Through Deep Learning (OTS-DL) And High-Level Security

Muthakshi S[1*], and Mahesh K[2]
[1]Research Scholar, Department of Computer Applications, Alagappa University
Karaikudi, Tamilnadu, India
[e-mail: muthakshi.researchscholar@gmail.com]
[2] Professor, Department of Computer Applications, Alagappa University
Karaikudi, Tamilnadu, India
[e-mail : mahesh.alagappa@gmail.com]
[*]Corresponding author: Muthakshi S

Received December 5, 2022; revised February 10, 2023; accepted February 27, 2023; published April 30, 2023

Abstract

Cloud computing is a new technology that has adapted to the traditional way of service providing. Service providers are responsible for managing the allocation of resources. Selecting suitable containers and bandwidth for job scheduling has been a challenging task for the service providers. There are several existing systems that have introduced many algorithms for resource allocation. To overcome these challenges, the proposed system introduces an Optimized Task Scheduling Algorithm with Deep Learning (OTS-DL). When a job is assigned to a Cloud Service Provider (CSP), the containers are allocated automatically. The article segregates the containers as' Long-Term Container (LTC)' and 'Short-Term Container (STC)' for resource allocation. The system leverages an 'Optimized Task Scheduling Algorithm' to maximize the resource utilisation that initially inquires for micro-task and macro-task dependencies. The bottleneck task is chosen and acted upon accordingly. Further, the system initializes a 'Deep Learning' (DL) for implementing all the progressive steps of job scheduling in the cloud. Further, to overcome container attacks and errors, the system formulates a Container Convergence (Fault Tolerance) theory with high-level security. The results demonstrate that the used optimization algorithm is more effective for implementing a complete resource allocation and solving the large-scale optimization problem of resource allocation and security issues.

Keywords: Resource allocation, Containers, Micro-macro task, Optimized Job Scheduling Algorithm, Security.

1. Introduction

Generally, in cloud computing, cloud containers are a supporting technology for CSP and cloud users. Due to the utilization of containers, the cloud container has gained a high rate of growth online in recent years [1]. In order to account for the rise of cloud containers, some cloud orchestration engines, such as Docker, Mesosphere, and Kubernetes, as well as several other applications, are introduced in the cloud platform [2]. Cloud for resource allocation and business collaborations among the CSPs and users are made. Many cloud vendors such as Amazon, Google, and Microsoft contain globally distributed containers and can provide the required resources for workflows [3]. The container includes only programme tasks provided for necessary dependencies and task completion. A pay-per-use billing model governs the user's access. The model needs a proper scheduling theory to ensure cost-effective resource usage.

There is always a big debate on improvising the resource utilization of cloud containers, task scheduling, and security maintenance. Initializing a proper job schedule on cloud platforms is still a key concern for the cloud service provider (CSP). When a massive number of tasks are handled, a reasonable resource allocation is mandatory. Static scheduling is the basic, traditional job scheduling method that builds a bidding relationship between the node and the container. The bond between the container and the node is maintained until the required job is unloaded or completed. After completing the task, the resources utilized by the container can be reused for a new task. The general form of resource allocation that works in the cloud is produced. In the beginning, the user seeks container availability from the service provider. The space allocation for the application is checked, and a suitable container is selected and allocated for the resource.

The proposed theory is put into effect to address the performance issue with resource allocation and to keep things stable. The proposed system focuses on enhancing task completion, security, and the task scheduling of containerized cloud services. The main contribution of the paper is as follows:

1. The proposed system categorizes container selection as long-term container (LTC) and short-term container (STC) selection. When a user data entry appears, the CSP automatically seeks container availability. According to the resource space, the container is allocated. The user jobs are allocated to short-term containers. The most frequent jobs are automatically assigned to long-term containers.
2. The system uses an ‘Optimized job scheduling algorithm' to analyse the complexities and produce allocations accordingly.
3. Furthermore, the system evaluates two dependency metrics before allocating any Container: 1) Micro Task Dependencies 2) Macro Task Dependencies. If the macro is dependent on the micro, then all the pending jobs in the micro are completed, and the macro job is performed. Likewise, the bottleneck dependency is completed first, and the job scheduling is continued according to the preferences.
4. Deep learning (DL) is used to improve task scheduling optimization. By using DNL, the security issues are found and rectified once detected.
5. A container split technique and searchable encryption technique are proposed to enhance resource allocation and job scheduling security.
6. Furthermore, the system experimentally verifies the accuracy of the system prototype at small scales and evaluates its efficiency of allocation at large scales.

The remainder of this paper, Section 2, discusses related works, the background of containerized cloud services, and the unique challenges. The paper further formulates the issues under different scenarios and analyses the efficiency of optimization job scheduling algorithms in Section 3. The proposed theory's implementation to justify the proposed algorithms' efficiency is discussed in Section 4, and experimental results are presented in Section 5. Section 6 discusses the work on job scheduling and concludes the paper.

2. Related work

This section reviews the current studies on Resource Allocation in Cloud Containers in terms of the issue model and methods to enhance the performance of container allocation and job scheduling metrics.

The researchers had much difficulty with optimizing job scheduling in the previous models. The works of various authors have been discussed, and assumptions have been discussed.

Zhang et al. [4] noted a resource allocation without the arrival and departure times, and the applications are held only for a particular period.

Many researchers describe how container placement is the most challenging part of resource allocation [5]. The paper examines the current container allocation and job scheduling models in terms of resource dimension, cost, and constraints. Existing studies for the schedule primarily have two objectives. The primary one is from the perspective of cloud service providers. CSPs concentrate on reducing the energy consumption of the used PMs or increasing resource utilization.

Guan et al. [6] consider the energy consumption and the price of the information exchange between containers. Guerrero et al. [7] discuss resource optimization of containers in multi-cloud for micro-services-based applications, focusing on the value of the used resources and concentrating on the energy consumption of cloud data centers.

Abhishek Kumar et al. [8] proposed a hybrid data security scheme called the Improved Attribute-Based Encryption Scheme (IABES). This model used two encryption algorithms, namely the Advanced Encryption Standard (AES) and the Attribute-Based Encryption (ABE) algorithm. Due to the hybrid model, the data could be maintained on a cloud server with appropriate security measures.

Baldominos Gómez et al. [9] proposed to deploy various machine learning approaches that could be used to estimate the future price of Elastic Cloud Compute (EC2) at spot instances. The proposed model used different techniques for the linear, ridge, and lasso regressions. K-nearest neighbors, multilayer perceptrons, and random forests were used in the model.

Zhang et al. [10] developed a model that combines various technologies like cryptography, blockchain, and the Interplanetary File System (IPFS). The proposed method is a fine-grained and flexible terminal data access control scheme. The scheme was based on ciphertext-policy attribute-based encryption (CP-ABE).

Choudhary and Pahuja [11] proposed and developed a new Steering Convention for Vitality Effective Systems (SC-VFS) technique. The developed model could be used for detecting doppelganger attacks. The model could be used in IoT-based intelligent health applications, for instance, in a green corridor for transplant pushback. The model was advantageous as it improved vitality proficiency, which is a critical constraint in WSN frameworks.

2.1 Optimization Algorithms

Several algorithms and approaches are used to rectify the optimization issues in cloud allocation. Tan et al. [12] introduced a multi-objective NSGA-II to address the multi-objective optimization issue. The availability of applications and the energy consumption requirements of container-based clouds are checked and reported to the server. This may be effective but not time-efficient, as it requires checking availability.

Niu et al. [13] stated a workflow allocation model named "Geo-aware Multiagent Task Allocation Approach" that focuses on large-scale scientific workflow execution in optimizing container-based metrics. A Cooperative Co-evolution Genetic Programming hyper-heuristic model has been introduced to overcome the resource allocation container (RAC) problem [14]. The game theory focuses on developing cooperative and non-cooperative theoretic approaches to scheduling, selection, and communication.

Mei et al. proposed an energy-aware scheduling algorithm for sporadic tasks on the dvs platform [15]. Zhu et al. [16] introduced a self-adapting job scheduling algorithm and dynamic priority scheduling for resource management. Zho et al. [17] developed a Scheduling Framework for Cloud Container Services. Yin et al. [18] produced job scheduling and resource allocation in fog computing relating to containers for manufacturing. The introduction of an elasticity control mechanism for cloud containers [19]. A Resource and Context Aware Deployment of Containerized Micro-Services was introduced [20].

A locality-aware scheduling model with load balance and application ability is possible [21].

Peng Zhao et al. [22] proposed an integration model with a Multi-cloud Access Control Policy. The author uses an Attribute-based Evaluation Model, Four-value Logic with Extensiveness, and Four-value Logic Related Policy Integration Operators. Through this, the policy integration technique is made easy. To overcome the optimization issues, a GRA-based service-aware flow model (GRSA) has been proposed by WenlongKe et al. [23].

In [24], Zhao et al. receive proportional allocations and assign them to fair shares. On the other hand, clients allocated to different queues receive allocations that maximize resource utilization.

Zhang et al. [25] proposed a Private Key Generator (PKG) to handle large-scale users. Here, the base PKG represents the lower-level PKGs providing a proper auditing scheme.

2.2. Cloud Security Challenges

Cloud servers (CS) are not always trustworthy because they are not completely under the control of the public. Once a data owner uploads sensitive data to the cloud server, CS may not be able to control the full data as attacks or interruptions may occur. This may lead to malicious users, attackers, and cloud server providers (CSP) accessing or stealing owners' sensitive records or information. Consequently, the task of preserving each and every piece of data in cloud server environment have become much more important [26, 27]. Zhu et al. [28] addressed the certificate management issues and designed a searchable encryption scheme for proof identification.

Wu et al. [29] created a cryptography theme related to a certificate-less crypto-system. Then, the existing system constructed a unique CL-SPKE technique for issues in the keyword attack. By implementing a random oracle model, the system gains the capacity to resist keyword attacks.

S Muthakshi, K Mahesh, in [30] produced a Container selection (CS) process implementation to choose suitable containers. Further, this paper suggests that the system determines extensive neural learning (NL) in cloud services for cloud container selection. In [31], the system stated a profit/loss calculator calculating the allocation or transaction, where the containers are well-monitored and controlled. Further, a trustworthy container's rating, report, or feedback is validated and suggested to the tenant. Also, the recommendation process is handled as a cycle.

3. Method implementation

3.1 Optimized Task Scheduling with Deep Neural Learning (OTS-DL)

In the proposal, the system leverages a job scheduling mechanism named the Optimized Task Scheduling Model. When a user's data centers are used for storage, the CSP immediately enquires and checks for the suitability of the containerized cloud. To make the optimization process more evident, container segregation has been introduced. The segregation process helps find the optimal solution for resource allocation and suggests the exact match for resource storage in the container. The system studies task completion performance. Deep learning (DL) is used to choose the optimal solution and perform secure resource allocation.

In the proposed cloud execution model, the cloud service provider (CSP) chooses and prioritizes the tasks according to their dependencies while assigning them. Generally, the tasks provided to the service provider are stated as ‘micro-task’ and ‘macro-task’. The dependencies between the tasks are found and prioritized accordingly.

- Micro-Task: The small tasks are handled by Micro-tasks. Micro-tasks are completed before committing to larger tasks. For instance, checking the resource origin, size, energy consumption, resource utilization, replicas, type of the resource, etc.
- Macro-Task: Large size tasks are assigned to Macro-tasks for handling big resources. The system proposes and optimized job scheduling, resource allocation, etc.,

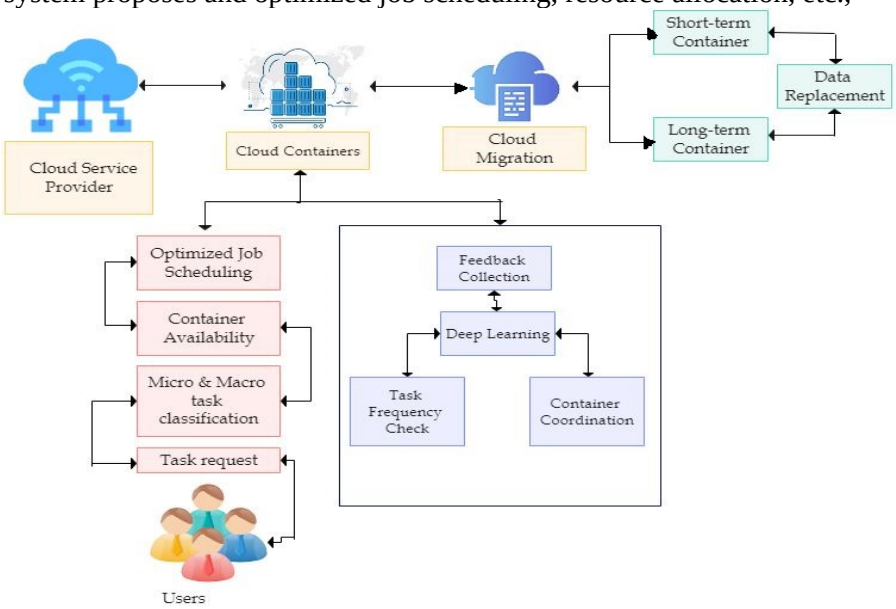

Fig. 1. Proposed System Architecture

There are chances that the macro-task may be dependent on the micro and vice versa. So, the bottleneck task dependency is checked, and priority is given to the required task at the required time. The bottleneck tasks are collected by initializing some classifiers with the help of neural learning. Then, the priority task is found and provided to the container for proper completion. For this execution, temporary containers can be hired, or long-term containers can be utilized for task completion. The above **Fig. 1** demonstrates the proposed architecture, starting from the user requesting CSP for job scheduling and resource allocation. Then, the cloud service provider (CSP) classifies the resources and performs job scheduling using deep learning (DL). Furthermore, a job scheduling optimizer is used to optimize the jobs and assign tasks accordingly. The cloud container is segregated into long-term containers (LTC) and short-term containers (STC) and performs allocation by resource classification and size. While using long-term containers, there are chances of flow leakage, leading to some major security issues.

3.2 Task Scheduling in Container

The system follows container segregation for task scheduling and allocation. Here, the system introduces two types of containers: Short-term containers (STC) and Long-term containers (LTC).

3.2.1 Short Term Container (STC)

- Short-term containers follow normal job scheduling metrics to provide an ‘optimized container allocation’.
- Generally, in a cloud execution model, CSP provides allocation for the new user data. When a resource is entered for allocation, the service provider immediately searches for availability in STC based on the size and preference of the resource.
- Initially, the STC verifies the need for long-term containers (LTC) through machine learning (ML). If the task needs a long-term container, then immediately, using container switching, it switches to LTC and adapts the required space for allocation.
- Entered jobs are assigned to STC for job completion.
- STC checks for the performance, security, and trustworthiness of the user and performs allocation.

3.2.2 Long-Term Container (LTC)

- Initially, the LTC coordinates the user and the container to perform the job scheduling. After coordination, the ‘container migration’ is enabled, where the resources are validated and allocated to the users’ requirements and container availability.
- Generally, the CSP contains some retained clients or users' jobs for resource allocation. Regular users are retained, and the users are mentioned as long-term users. These types of client resources and jobs are selected for long-term container (LTC) storage. The container undergoes several optimization processes for selection.
- Once a container is chosen for the task process, CSP retains that container for long-term dealing with user job scheduling.
- In case of any middle breakage from the user side or from the container utilization side, immediately without any hesitation, the progress is withdrawn within the minimum

notice period.

- Due to the withdrawal concept, both the container and the user attain more profit. The main advantage of LTC is minimal search time due to pre-determined allocation. LTCs are long-term containers used for allocating frequent jobs.
- CSP to allocate the correct container for the resources.

Table 1. Proposed Results

Job	Processing time(ms)	Slack time(ms)	Container switching time	Process Completion delay	Heuristics	Current completion time
1	1000	5000	0	0	1000	1000
2	35000	175000	5000	0	40000	38000
3	50000	250000	5000	0	60000	50000
4	25000	125000	5000	0	0	30000
5	10000	50000	5000	0	0	15000

3.2.3 Feedback from both the container and the user

Finally, feedback from the container side and user side to improve the implementation is enabled. After completion of every task scheduling process, feedback is regularly collected from both the user and the container. The feedback contains details about the user satisfactory level (container, cost-efficiency, and speed) and the container satisfactory level (cost, job size, etc.).

Furthermore, the LTC is provided with levels that are capable of container extension and shrinking according to the task and resource allocation. When more utilization is needed, the container extension is used. The system goes with container shrinking when less utilization is needed. Hence, LTC is proved to be more useful and effective for proper ‘optimized job scheduling’.

Table 2. Container convergence

Container	Container clearance	Data replacement	Security monitoring	Container convergence
1	100	1000	0	0
2	100	5000	0	0
3	100	5000	0	0
4	100	60000	1	1
5	100	20000	0	0

From the above (**Table 1**) The jobs are initially assigned, and the processing time (ms) is set. The amount of setup time for tasks is defined as slack time, which can be delayed without causing an additional task. The container switching time is set, and the switching time is calculated. Heuristics are located according to the job schedule, and the current completion time is arranged accordingly.

In **Table 2**, container clearance is done when it reaches 100. When the container limit exceeds the data, it is replaced by the higher-level containers according to the data size. This term is defined as "Container Convergence." The data in the containers is assigned, and proper security monitoring is ensured.

Algorithm 1. Implementation

```
Begin
{
 Read: sequential jobs
 Set: i=0// assign starting job for process
 Initialize: container =0 //initialize container for job scheduling
Do
{
   While (cache available)
      {
        user and container subscription
        optimized job scheduling
        container switching
        container clearance
        deep learning of feedback
              If container usage greater than threshold
                   user container compatibility check
                      If compatible then map container
                         learn user and container usage
                      End if
              End if
        }
} Until (EOF)
}
```

Step 1: The jobs are provided as input. Then, the jobs are processed in a sequential manner.

Step 2: The jobs are assigned for container allocation.

Step 3: Then, the containers are set for performing the job scheduling process.

Step 4: The UC user container subscription is completed, and job scheduling is optimized**.**

Step 5: According to the container size, container switching is performed Cs. After container retrieval, a proper container clearance Cc is done.

Step 6: The feedback from users and container providers is collected using deep learning.

Step 7: When the container usage is greater than the threshold (Cu > T), a compatibility check is permitted.

Step 8: The user container usage and compatibility are checked on a regular basis, and the appropriate allocation is carried out.

Algorithm 2. Algorithm for Short-Term and Long-Term Container.

Input: Input data allocation
Output: Allocation with efficient container Convergence

```
Begin
{
 Read: sequential jobs
 Set: i=0// assign starting job for process
 Initialize: container =0 //initialize container for job scheduling
Do
{
   While (job frequency)
      {
         Check for micro or macro task
              If frequency greater than threshold
                   Allocate long term container
               Else
                    Allocate short term container
      }
} Until (EOF)
}
```

Algorithm 2. Implementation

Step 1: In algorithm 2, the job scheduling is processed considering the user data size.

Step 2: The jobs are assigned for container allocation.

Step 3: Then, the containers are set for performing the job scheduling process.

Step 4: Then the frequency of the job is noted. The most repeated and preferable jobs are segregated.

Step 5: The frequent jobs are checked for micro or macro tasks.

Step 6: The most frequent jobs are allocated to long-term containers, considering the macro and micro tasks.

Step 7: The non-frequent tasks are allocated to the short-term container itself.

Algorithm 3. Defense algorithm through container convergence

Input: Input data allocation
Output: Allocation with efficient container Convergence

```
Begin
{
 Read: sequential jobs
 Set: i=0// assign starting job for process
 Initialize: container =0 //initialize container for job scheduling
Do
{
   While (container monitoring)
      {
              If congestion occurs
                  {
                   If attack detected
                      Security container convergence
                   else
                      process container convergence
                    }
                    Container recovery
              End if
      }
} Until (EOF)
}
```

Algorithm 3: Implementation

In the above algorithm, the security measures are processed when the container convergence method is opted for.

Step 1: In algorithm 3, the job scheduling is processed considering the user data size.

Step 2: The jobs are assigned for container allocation.

Step 3: Then, the containers are set for performing the job scheduling process.

Step 4: Then, the containers with jobs are monitored regularly.

Step 5: If any inconvenience or congestion occurs, the security system is activated.

Step 6: When an attack is detected, fault tolerance and the ECC technique are selected.

Step 7: Ensure secure user data allocation and proper container allocation.

3.3 High-level Security

The goal of the system is to provide a high-level of security for containers in the cloud. The previous method has been adapted for several security-based techniques. But handling the major attacks, like DoS attacks, is always a challenging task. A denial-of-service (DoS) attack is very dangerous and causes loads of network traffic by crashing services.

Reputation by container Robustness

Considering the storage capacity of the user, the containers are assigned and allocated. Some attackers perform fraudulent activities by injecting threats or unwanted data into the middle of the execution. The undesirable data entry is unknown to both the container and the user. The container, as usual, accelerates the capacity and occupies maximum storage. The fraud activity spoils the reputation of the cloud container and the CSP that recommends the container to the user. Hence, the relationship between the CSP and the user is affected.

To overcome all the above-mentioned security issues, the system has proposed a new high-level security theory, adopting searchable encryption and fault tolerance. Generally, there are several techniques to encrypt and store data in a container. But maintaining the data and producing it to the owner without any interruption or attack has become more challenging. To resolve the issue, the system has implemented a high-level security deal to handle DoS attacks.

The following stages are followed to handle the unwanted blockage from several attacks:

a) Container Split Technique

To enhance security, the system uses encryption and container split technique. To divert the attackers' attention and destroy the pattern flow, the system uses multiple containers for job and resource allocation.

If a job needs high-end security, then a swap technique is used that performs a job scheduling swap inside the container. To provide more confidentiality for each container, various types of encryption and decryption combinations of algorithms are used.

b) Cache Memory Maintenance for Fault Tolerance

At this stage, each container is maintained with a cache memory space. When an attack or error enters and creates any unwanted traffic for space occupation, the cache memory space is used to evaluate the unwanted data, predict the source of the error, and block it immediately.

c) Security Monitoring

At this stage, the incoming and outgoing data have been well examined. The origin of the user is checked to see if the data is from a trustworthy owner or not. By using the searchable encryption method, the authorized users are matched, and the task scheduling encryption and decryption are performed. The details of the data are monitored, collected, and reported continuously. If any abnormal activity is detected, the entire data valuation from the source and their inter-connected networks are also verified. If any kind of disruption is noted in the

source networks, then that particular connection is disconnected, and the other task scheduling work is resumed.

d) Container Convergence

The Container Convergence method is chosen to eliminate the attacks. In the ongoing process, if any type of attack is detected, the other containers converge and form a strong bond to overcome the particular attack. Further, by initiating a fault tolerance method, any type of error or attack can be detected and rectified easily. Finally, by implementing the above-mentioned methods, the security is maintained, and data can be securely recovered from the container.

e) Fake Value Replacement

Finally, the original data in the container is replaced with some other random fake values, and then the data clearance is performed. When attackers try to recover the user data, only the faked and replaced values can be retained, not the original data. This enhances security at a higher level.

4. Result analysis

In the result analysis, a comparison of all the existing algorithms with the proposed model is done.

A multimedia dataset from Kaggle is taken and produced by Jaafar Bendriss. A deep learning-based SLA management for NFV-based services.

Forecasting and anticipating SLO breaches in programmable networks are used. The system uses cognitive 5G networks: comprehensive operator use cases with machine learning for management operations.

In **Fig. 2**, the comparison of existing algorithms, such as the PSO approach and locality aware scheduling, is done with the proposed OTS-DL algorithm. PSO is the Particle Swarm Optimization (PSO) used for optimizing the job, which is not efficient. In the comparison, OTS-DL From **Fig. 3**, the overall comparison of the proposed algorithm (OTS-DL) with the existing algorithms such as multi-objective NSGA-11, self-adapting, and ant colony algorithms is made.

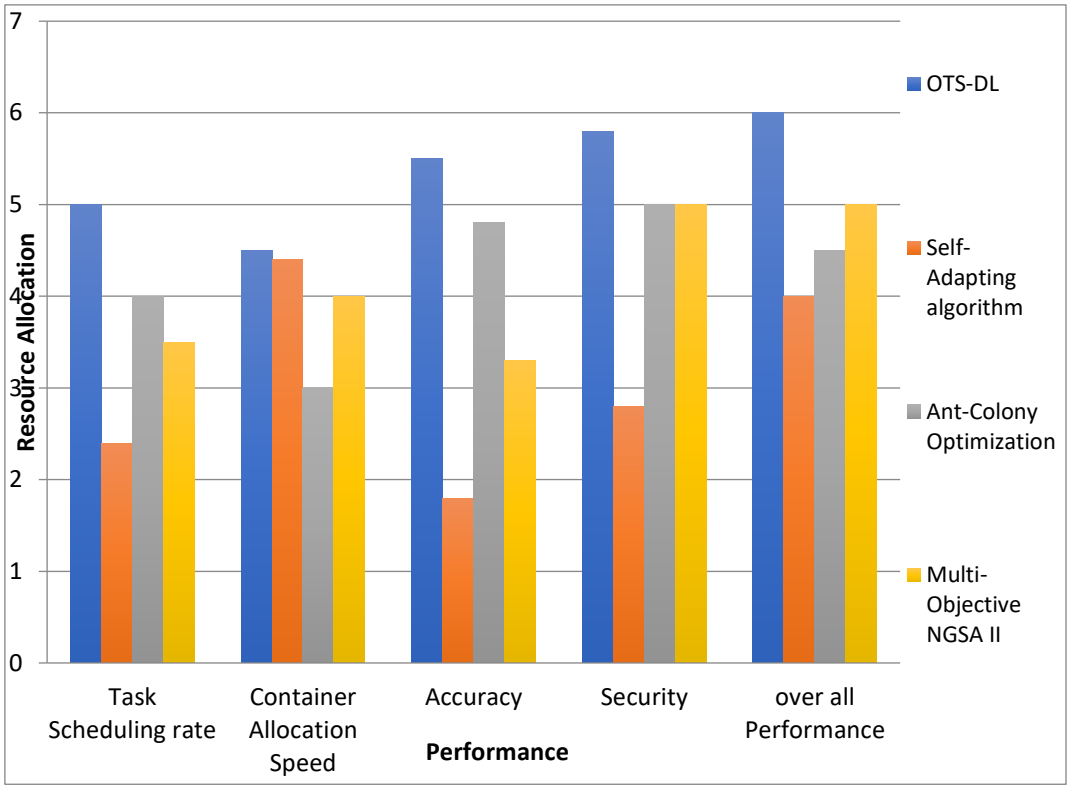

Fig. 2. Existing vs. Proposed System

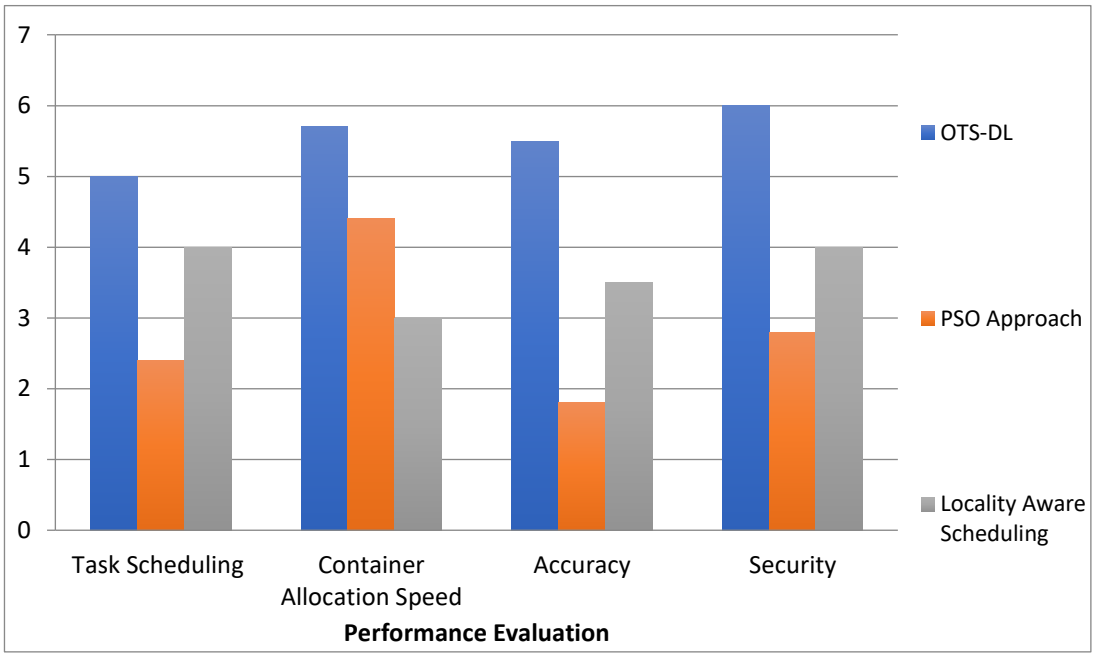

Fig. 3. Existing vs. Proposed Algorithm

The proposed algorithm, OTS-DL, is proven to be the optimal solution in terms of accuracy, allocation speed, security, and overall performance.

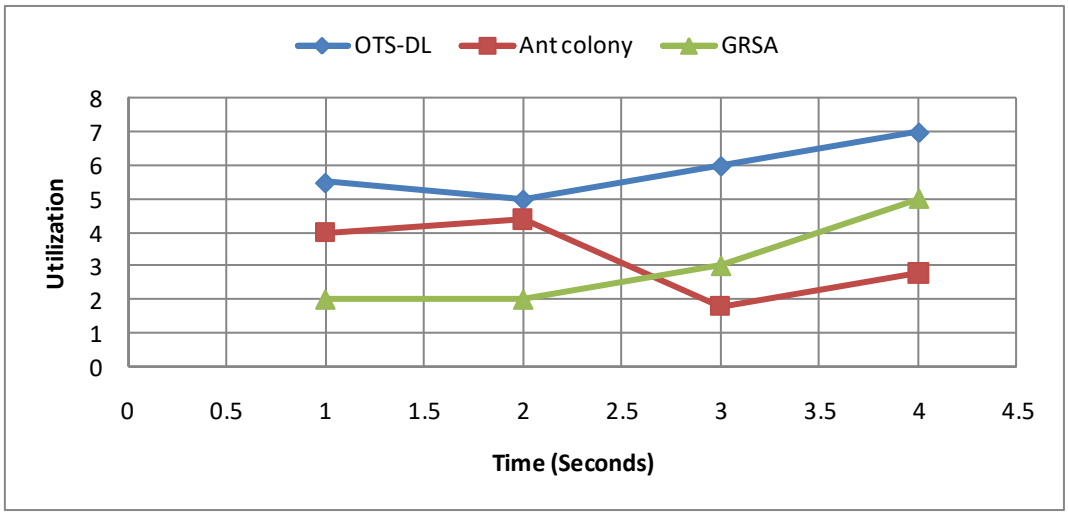

Fig. 4. System Utilization Existing vs. Proposed

In the above graph, **Fig. 4**, the utilization rate is shown. GRSA is a service-aware flow scheduler for cloud storage used in data centre networks that lack time and speed. The time interval between the existing and proposed systems is being deliberated.

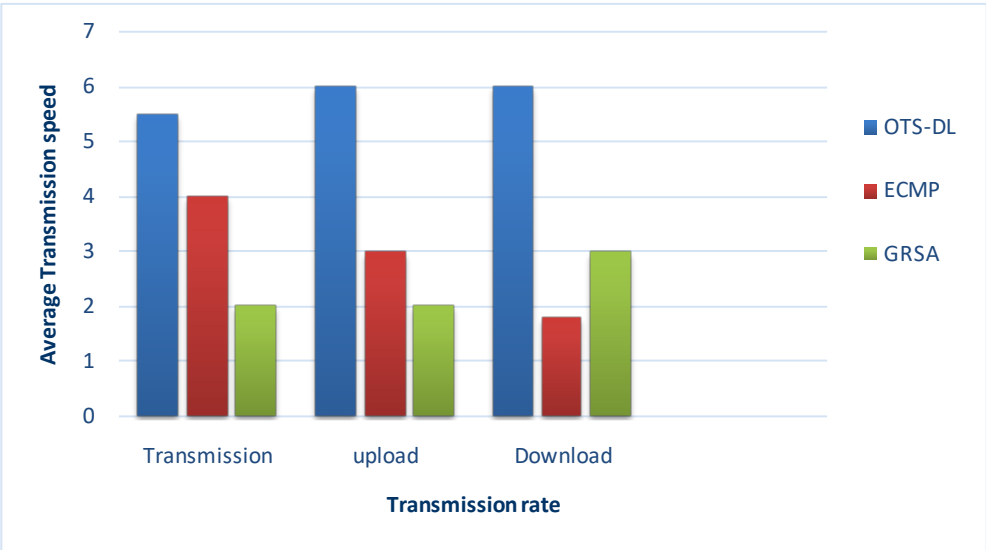

Fig. 5. Average transmission speed Existing vs. Proposed

In **Fig. 5**, the average transmission speed is estimated. Here the transmission speed, upload time, and download time are compared. The proposed system proves to be faster.

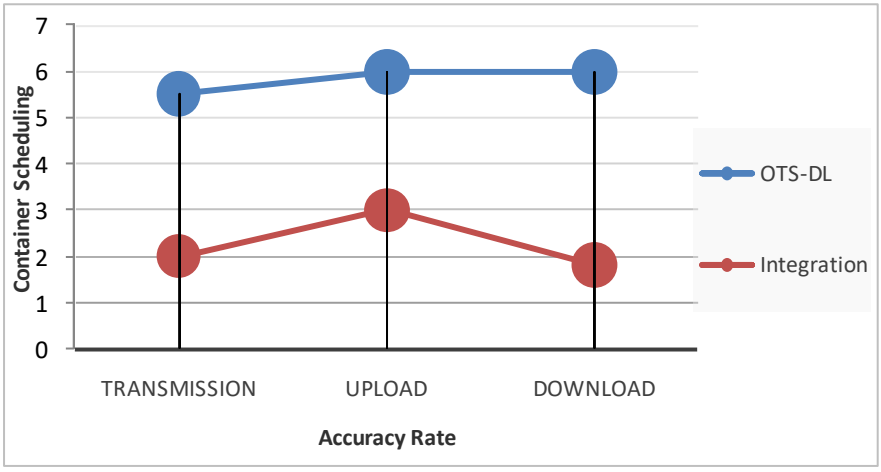

Fig. 6. Average Container Scheduling Existing vs. Proposed

In **Fig. 6**, the scheduling and accuracy rates are compared, and, from the comparison with the integration model, the proposed OTS-DL model proves to be better in accuracy.

5. Conclusion

In the cloud, job scheduling and resource allocation are the most challenging tasks to be handled by the cloud service provider. The paper introduces a new novel theory known as the Optimized Task Scheduling Algorithm via Deep Learning (OTS-DL). The algorithm segregates the containers into long-term and short-term containers and allocates the tasks according to the availability of the containers. Simultaneously, the micro-task and macro-task dependencies are classified using deep neural learning, and priority is provided accordingly. Moreover, container split and searchable encryption security monitoring are used to enhance security and provide high confidentiality for the resources. The proposed method provides an optimal solution in terms of the parameters of accuracy of the operations, allocation speed of the resources, security, and overall performance. The transmission rate is high when compared to the existing methods. Hence, the method maintains the efficiency, speed, and accuracy of task scheduling and resource allocation. In conclusion, the overall performance can be increased by using the proposed method, thereby facilitating the scientific community in various ways like power consumption, etc. The proposed method has been deployed in a contained environment. There may be a certain challenge when the method is deployed in a real time environment. In the future, security measures will be tested in real-time applications, and more optimized algorithms will be used to attain greater speed.

Acknowledgement

This research work has been supported by University Grants Commission, New Delhi, India. The authors are thankful to the anonymous reviewers whose comments helped to enhance this paper.

References

[1] M. Lin, J. Xi,W. Bai, and J.Wu, “Ant colony algorithm for multi-objective optimization of container-based microservice scheduling in cloud," *IEEE Access*, vol. 7, pp. 83088-83100, 2010. Article (CrossRef Link)

[2] R. Tang, “Research on resources scheduling strategy of container cloud platform based on Kubernetes," M.S. thesis, Dept. Elect. Eng., UESTC Univ., Si Chuan, China, 2017.

[3] A. C. Zhou, B. He, X. Cheng, and C. T. Lau, “A declarative optimization engine for resource provisioning of scientific workflows in geo-distributed clouds,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 3, pp. 647–661, 2017. Article (CrossRef Link)

[4] X. Zhang, T. Wu, M. Chen, T. Wei, J. Zhou, S. Hu, and R. Buyya, “Energy-aware virtual machine allocation for cloud with resource reservation,” *Journal of Systems and Software*, vol. 147, pp. 147–161, 2019. Article (CrossRef Link)

[5] Wolke, M. Bichler, and T. Setzer, “Planning vs. dynamic control:Resource allocation in corporate clouds,” *IEEE Transactions on Cloud Computing*, vol. 4, no. 3, pp. 322–335, 2016. Article (CrossRef Link)

[6] X. Guan, X. Wan, B. Y. Choi, S. Song, and J. Zhu, “Application Oriented Dynamic Resource Allocation for Data Centers Using Docker Containers,” *IEEE Communications Letters*, vol. 21, no. 3, pp. 504–507, 2017. Article (CrossRef Link)

[7] C. Guerrero, I. Lera, and C. Juiz, “Resource optimization of container orchestration: a case study in multi-cloud microservicesbased applications,” *Springer The Journal of Supercomputing*, vol. 74, no. 7, pp. 2956—2983, 2018. Article (CrossRef Link)

[8] A. Kumar, S. A. Kumar, V. Dutt, A. K. Dubey, S. Narang., “A Hybrid Secure Cloud Platform Maintenance Based on Improved Attribute-Based Encryption Strategies,” *International Journal of Interactive Multimedia and Artificial Intelligence*, 2021. Article (CrossRef Link)

[9] Baldominos Gómez, A., Saez, Y., Quintana, D., & Isasi, P., "AWS PredSpot: Machine Learning for Predicting the Price of Spot Instances in AWS Cloud," *International Journal of Interactive Multimedia and Artificial Intelligence*, Vol. 7, No. 3, 2022. Article (CrossRef Link)

[10] Zhang, G., Chen, X., Zhang, L., Feng, B., Guo, X., Liang, J., & Zhang, Y., “STAIBT: blockchain and CP-ABE empowered secure and trusted agricultural IoT blockchain terminal,” *International Journal of Interactive Multimedia and Artificial Intelligence*, Vol. 7, No. 5, 2022. Article (CrossRef Link)

[11] Choudhary, D., & Pahuja, R, “Improvement in quality of service against doppelganger attacks for connected network,” *International Journal of Interactive Multimedia and Artificial Intelligence*, Vol. 7, No. 5, 2022. Article (CrossRef Link)

[12] B. Tan, H. Ma and Y. Mei, "A NSGA-II-based Approach for Multi-objective Micro-service Allocation in Container-based Clouds," in *Proc. of 2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, pp. 282-289, 2020. Article (CrossRef Link)

[13] M. Niu, B. Cheng, Y. Feng and J. Chen, "GMTA: A Geo-Aware Multi-Agent Task Allocation Approach for Scientific Workflows in Container-Based Cloud," *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, pp. 1568-1581, Sept. 2020. Article (CrossRef Link)

[14] B. Tan, H. Ma, Y. Mei and M. Zhang, "A Cooperative Coevolution Genetic Programming Hyper-Heuristic Approach for On-line Resource Allocation in Container-based Clouds," *IEEE Transactions on Cloud Computing*, vol. 10, no. 3, pp. 1500-1514, 2022. Article (CrossRef Link)

[15] J. Mei, K. Li, J. Hu, S. Yin, and E. H.-M. Sha, “Energy-aware preemptive scheduling algorithm for sporadic tasks on DVS platform,” *Microprocessors and Microsystems*, vol. 37, no. 1, pp. 99 – 112, 2013. Article (CrossRef Link)

[16] L. Zhu, K. Huang, Y. Hu and X. Tai, "A Self-Adapting Task Scheduling Algorithm for Container Cloud Using Learning Automata," *IEEE Access*, vol. 9, pp. 81236-81252, 2021. Article (CrossRef Link)

[17] R. Zhou, Z. Li and C. Wu, "Scheduling Frameworks for Cloud Container Services," *IEEE/ACM Transactions on Networking*, vol. 26, no. 1, pp. 436-450, Feb. 2018. Article (CrossRef Link)

[18] L. Yin, J. Luo and H. Luo, "Tasks Scheduling and Resource Allocation in Fog Computing Based on Containers for Smart Manufacturing," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4712-4721, Oct. 2018. Article (CrossRef Link)

[19] W. A. Hanafy, A. E. Mohamed and S. A. Salem, "A New Infrastructure Elasticity Control Algorithm for Containerized Cloud," *IEEE Access*, vol. 7, pp. 39731-39741, 2019. Article (CrossRef Link).

[20] H. Sami, A. Mourad and W. El-Hajj, "Vehicular-OBUs-As-On-Demand-Fogs: Resource and Context Aware Deployment of Containerized Micro-Services," *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 778-790, April 2020. Article (CrossRef Link).

[21] D. Zhao, M. Mohamed and H. Ludwig, "Locality-Aware Scheduling for Containers in Cloud Computing," *IEEE Transactions on Cloud Computing*, vol. 8, no. 2, pp. 635-646, 1 April-June 2020. Article (CrossRef Link)

[22] P. Zhao, L. Wu, Z. Hong and H. Sun, "Research on multicloud access control policy integration framework," *China Communications*, vol. 16, no. 9, pp. 222-234, Sept. 2019. Article (CrossRef Link)

[23] W. Ke, Y. Wang and M. Ye, "GRSA: Service-aware flow scheduling for cloud storage datacenter networks," *China Communications*, vol. 17, no. 6, pp. 164-179, June 2020. Article (CrossRef Link)

[24] L. Zhao, M. Du and L. Chen, "A new multi-resource allocation mechanism: A tradeoff between fairness and efficiency in cloud computing," *China Communications*, vol. 15, no. 3, pp. 57-77, March 2018. Article (CrossRef Link)

[25] Y. Zhang, H. Zhang, R. Hao and J. Yu, "Authorized identity-based public cloud storage auditing scheme with hierarchical structure for large-scale user groups," *China Communications*, vol. 15, no. 11, pp. 111-121, Nov. 2018. Article (CrossRef Link)

[26] D. Alsmadi and V. Prybutok, "Sharing and storage behavior via cloud computing: Security and privacy in research and practice," *Comput. Hum. Behav.*, vol. 85, pp. 218-226, Aug. 2018. Article (CrossRef Link)

[27] Y. Li and C. G. Ma, "Review of research progress on searchable encryption," *J. Netw. Inf. Secur.*, vol. 4, no. 7, pp. 13-21, 2018.

[28] M. Zhu, Y. Chen, and Y. Hu, "Identity-based searchable encryption scheme supporting proxy re-encryption," *Comput. Eng.*, vol. 45, no. 1, pp. 129-135, 2019. Article (CrossRef Link)

[29] T. Y. Wu, C. M. Chen, K. H. Wang, C. Meng, and E. K.Wang, "A provably secure certificateless public key encryption with keyword search," *J. Chin. Inst. Eng.*, vol. 42, no. 1, pp. 20-28, 2019. Article (CrossRef Link)

[30] S Muthakshi, K Mahesh, "Retracted: Container selection processing implementing extensive neural learning in cloud services," *Materials Today: Proceedings*, 2021. Article(CrossRef Link)

[31] M. S and M. K, "Enhanced Cloud Optimization Model for CSP, Tenant and User Through Container," in *Proc. of IEEE International Conference on Mobile Networks and Wireless Communications (ICMNWC)*, pp. 1-5, 2021. Article (CrossRef Link)

S.Muthakshi received the M.Phil degree in Computer Science from Alagappa University, Karaikudi, India in 2013. Currently, she is pursuing her Ph.D degree in the Department of Computer Applications, Alagappa University. Her research interest includes Cloud Computing, Cloud Container security.

Dr. K.Mahesh received the Ph.D degree in Computer science from Alagappa University, India in 2012. He has 32 years of experience in the Department of Computer Applications, Alagappa University, Karaikudi, India. His research interest includes Video Processing Image Processing, and Cloud Security.