International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
DOI: https://doi.org/10.17762/ijritcc.v11i9.8114
***Article Received:** 07 July 2023 **Revised:** 28 August 2023 **Accepted:** 16 September 2023*

NASLMRP: Design of a Negotiation Aware Service Level Agreement Model for Resource Provisioning in Cloud Environments

Pallavi Shelke[1], Dr.Rekha Shahapurkar[2]
[1]PhD scholar student, CSE deartment , Oriental University,
Indore,Madhya Pradesh,India
pallavishelke12@gmail.com
[2]Associate professor ,CSE deartment , Oriental University,
Indore,Madhya Pradesh,India
rekhashahapurkar@gmail.com

Abstract—Cloud resource provisioning requires examining tasks, dependencies, deadlines, and capacity distribution. Scalability is hindered by incomplete or complex models. Comprehensive models with low-to-moderate QoS are unsuitable for real-time scenarios. This research proposes a Negotiation Aware SLA Model for Resource Provisioning in cloud deployments to address these challenges. In the proposed model, a task-level SLA maximizes resource allocation fairness and incorporates task dependency for correlated task types. This process's new tasks are processed by an efficient hierarchical task clustering process. Priority is assigned to each task. For efficient provisioning, an Elephant Herding Optimization (EHO) model allocates resources to these clusters based on task deadline and make-span levels. The EHO Model suggests a fitness function that shortens the make-span and raises deadline awareness. Q-Learning is used in the VM-aware negotiation framework for capacity tuning and task-shifting to post-process allocated tasks for faster task execution with minimal overhead. Because of these operations, the proposed model outperforms state-of-the-art models in heterogeneous cloud configurations and across multiple task types. The proposed model outperformed existing models in terms of make-span, deadline hit ratio, 9.2% lower computational cycles, 4.9% lower energy consumption, and 5.4% lower computational complexity, making it suitable for large-scale, real-time task scheduling

Keywords-Resource Provisioning, SLA Negotiation, EHO, VM, Task Shifting, QoS.

I. INTRODUCTION

Resource provisioning is a multidomain task that involves analysis of task sets in terms of their dependency levels, make-spans, deadline & originating sources, and mapping these tasks with cloud VMs via capacity optimizations. Researchers have proposed a broad range of models with distinct qualitative and quantitative performance features. A general-purpose SLA based task scheduling model that combines QoS aware stations (QST), and task make-span levels [1] is depicted in figure 1, wherein task dependencies are used to map different resource types.

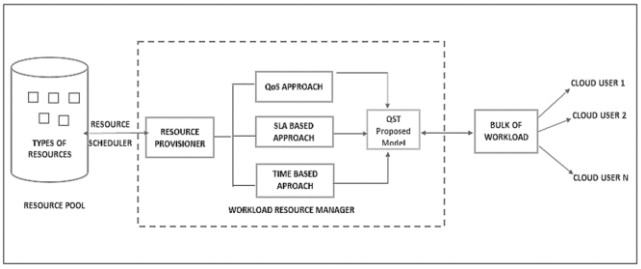


Fig 1: A typical SLA based resource provisioning model for different task types

Such models evaluate task requirements via following equation 1, wherein task delay, make-span, and other parameters are considered for comprehensive representation of different task types.

$$TR_j = \sum_{i=1}^{N_t} f\left(M_i, B_i, D_i, \sum_{k=1}^{N_t} Dep_{i,k}\right) \dots (1)$$

Where, TR_j represents task requirement for j^{th} task out of N_t tasks, while M, B, D represents internal task characteristics in terms of its Memory, Bandwidth & Delay requirements, $Dep_{i,k}$ represents dependency information between i^{th}&k^{th} tasks, and f represents a function to evaluate task requirements w.r.t. different deployment scenarios.Similar to this, the capacity of different resources is evaluated via equation 2, which aims at representing VMs capability to perform task scheduling process with optimum efficiency levels.

$$C = \sum_{i=1}^{N(VM)} f(MIPS_i, PE_i, B_i, M_i) \dots (2)$$

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
DOI: https://doi.org/10.17762/ijritcc.v11i9.8114
***Article Received:** 07 July 2023 **Revised:** 28 August 2023 **Accepted:** 16 September 2023*

Where, $N(VM)$ represents number of VMs available, while $MIPS \& PE$ expresses the Virtual Machine's capacity as millions of instructions per second (MIPS)., and For scheduling reasons, the number of Processing Elements that are included with the VM.

The behavior of a system can influences by a variety of external circumstances in a cloud computing setting. Violations of service level agreements (SLAs) are a major part of cloud computing since they reduce customer satisfaction and disrupt cloud providers, which could lead to penalties. Violations A service level agreement (SLA) may have happened if any of the following conditions apply: performance is supplied below the agreed-upon level; services are offered at proper level on the other hand with considerable delay; utilizations VM resource varies; or services are not provided at all. One of the most important parts of using services is negotiating Service Level Agreements (SLAs). Consumers and service providers form an alliance throughout the negotiating process. Convergence during settlement is influenced by the extent to which information may be shared throughout this procedure. The current negotiation process defines the number of parties involved, their respective roles and duties, the visibility of the agreements reached, the parameters of each individual negotiation session, and so on. SLA negotiation is an essential method for preservingthe integrity of cloud services and fostering trust between cloud users and suppliers. Cloud parties can establish Quality of Service (QoS) expectations for mission-critical service-based operations by outlining them in a Service Level Agreement (SLA).

Drawing from these considerations, researchers have developed different mapping models for provisioning resources to multiple task types. The next section of this text provides a survey of deployment-specific details, application-specific benefits, functional restrictions, and potential future applications for these models [2][3][4]. This review reveals that current provisioning models are either incomplete or excessively complex, thus restricting their scalability. Moreover, most of the comprehensive models showcase low-to-moderate levels of QoS (Quality of Service), thus cannot be used for real-time situation. To get around these limitations, Section 3 examines a novel's design of Negotiation Aware Service Level Agreement (SLA) Model for Resource Provisioning for cloud deployments. The examination of the proposed model was undertaken with regard to make-span, deadline hit ratio, computational cycles, energy consumption, and computational complexity in section 4, where it was also compared with several up-to-date versions. This will assist readers in validating model's performance under different use cases. In other words, the text concludes by offering insights that are relevant to specific contexts and deployment scenarios for the proposed model. Additionally, it suggests that fusion techniques can be applied to enhance the model's performance in a range of scenarios.

II. LITRATURE REVIEW

The researchers provide a broad range of various task scheduling models, each with its unique set of internal working components. For instance, research from[5][6] suggests using geo-distributed data analytics and a self-adaptive task scheduling model to estimate high-density data patterns while mapping workloads to various cloud architectures. This strategy, however, cannot be used to a variety of tasks since it is not scalable. The work that is suggested in[7] recommends employing Multiple Device Co-Processing of Data-Parallel Kernels to make it easier to apply the model for task scheduling in scattered settings, overcoming this challenge. To streamline the process of implementing the model and reduce the required time and effort, this is done. Due to this model's potential to predict workload patterns, capacity pre-emption across all VM types is enhanced. The Joint Task Scheduling and Containerizing (JTSC) model, the Genetic Algorithm with Mobility aware Task Scheduling (GAMTS) model, and the Deep Neural Network Scheduling (DNNS) model are just a few of the similar models that are examined in[8][9], among others. The aim of each of these models is to estimate a broad range of job kinds in real-time settings. These models allow for the uncomplicated implementation of scheduling methods for large-scale applications. The Whale Optimization Algorithm (WOA), energy, time, and rental cost (ETRC) optimization, and non-pre-emptive stochastic co-flow scheduling (NPSCS) are all approaches that are explored in[10][11][12][13] as strategies to enhance these models' performance on related sets of tasks. Non-pre-emptive stochastic co-flow scheduling is abbreviated as NPSCS, while energy, time, and rental cost optimization is abbreviated as ETRC. These models perform admirably in scenarios of low complexity, making them appropriate for use in reducing the detrimental impacts of scheduling errors in such circumstances. The fact that they have this quality makes them acceptable for use in large scheduling projects.

Elastic Task Scheduling Scheme[19] , Multi-Task Deep Reinforcement Learning (MTDRL)[16], Decomposition-based novel multi-objective evolutionary algorithm[17], "Energy-Efficient Scheduling (EES)[14] ، Spatial Scheduling That Considers Profit[15], Scaling of Dynamic Voltage and Frequency[18] and Novel Multi Objective Evolutionary Algorithm based on These models are all available. These models, however, are useless for dynamic task sets. The work in[20] recommends using a Dynamic and Resource Aware Load Balanced Scheduling Model (DRALBM) to help

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
DOI: https://doi.org/10.17762/ijritcc.v11i9.8114
***Article Received:** 07 July 2023 **Revised:** 28 August 2023 **Accepted:** 16 September 2023*

enhance their performance in situations where jobs are continually moving, and as a consequence, they are further enlarged. "Task Scheduling and Microservices based Computational Offloading" (TSMCO), Deep Neural Networks (DNN), and Energy-efficient Dynamic Scheduling Scheme (EDSS) are all proposed in [21][22][23] tofurther optimize their performance by incorporating high-density feature extraction under real-time scheduling scenarios. This is carried out to raise their general effectiveness. The complexity exhibited by these models renders them unsuitable for time-sensitive scenarios. Researchers in[24][25][26]suggest Parallel Processing, Deep Reinforcement Learning (DRL), and Earliest Deadline First (EDF) as scheduling techniques that are appropriate for high-velocity applications as a solution to this problem. The sources provided below have information on these techniques. Improved real-time functionality forthese models may be achieved by using processing methods of low complexity. The authors discuss methods to reduce the costs of performing interdependent task sets in[27][28][29][30][31]by combining task duplication, particle swarm optimization, and heuristics that account for unoccupied time slots (PSO). Contrarily, these models either cannot be deployed in a manner that makes them appropriate for use with real-time cloud jobs or they disobey the requirements for tasks and virtual machines (VM). Due to their extreme sensitivity to their surroundings, these models are also undesirable for use in extensive scheduling projects. The current provisioning and scheduling models can be classified as either overly complex or incomplete in their consideration of multiple task and resource parameters for scheduling. As a result, their potential to scale under real-time use cases is limited. Additionally, the majority of thorough models exhibit low-to-moderate levels of QoS (Quality of Service), making them unsuitable for usage in real-time settings. Next section proposes design of a unique Negotiation Aware Service Level Agreement (SLA) Model for Resource Provisioning for cloud deployments to get over these drawbacks. Different parameters were considered to measure the performance of this model like scheduling scenarios & validated w.r.t. multiple state-of-the-art models, which will assist readers to deploy the proposed model for their functionality-specific use cases

III. PROPOSED NEGOTIATION-AWARE MODEL FOR SERVICE LEVEL AGREEMENTS IN CLOUD RESOURCE PROVISIONING (NASLMRP).

Reviewing the existing resource provisioning models revealed that they are either non-comprehensive or have higher complexity than required, which restricts their ability to scale. Additionally, the majority of these models exhibit low-to-moderate levels of QoS (Quality of Service), making them unsuitable for use in real-time scenarios. This section proposes a new Negotiation solution to tackle previously mentioned difficulties by aware Service Level Agreement (SLA) Model for Resource Provisioning in cloud deployments. Figure 2, visually represents the workflow of the proposed model, showcasing the initiation phase. This phase incorporates the implementation of a task-level Service Level Agreement (SLA) aimed at optimizing fairness in resource allocation. Additionally, it takes into account the interdependence of task types that are interconnected.

An effective hierarchical task clustering process is used to process the new set of tasks generated by this process. As a result, all tasks are clustered into priority-level groups. An Elephant Herding Optimization (EHO) model that takes task deadline & make-span levels into account, allots these clusters to resources for effective provisioning operations. Designing a fitness function that lowers make-span levels while maintaining higher deadline awareness is advised by the EHO Model, which assists in improving its scalability & performance under multiple task types.

The assigned tasks are post-processed using a VM-aware negotiation framework that employs Q-Learning for efficient capacity tuning and task-provisioning, enabling quicker task execution with lower overheads. In the interest of simplicity, process flow is divided into three distinct sub-modules, each of which is explained in a separate section of this document. This approach enables readers to deploy the modules partially or in their entirety, depending on their specific use cases.

A. Design of an SLA Model for task clustering with dependency awareness

Initially all tasks and their dependency constraints are evaluated by an SLA Model, which assists in resequencing of tasks to obtain higher efficiency levels. To perform resequencing, a make-span difference is evaluated via equation 3.1,

$$MSD_{i,i+1}(IP) = MS_{i+1}(IP) - MS_i(IP) \dots (3.1)$$

Where, $MSD_{i,i+1}(IP)$ represents Make Span Difference between task $i \& i+1$, which is coming from the same IP address. Based on this difference, an MSD Threshold is evaluated via equation 3.2,

$$MSD_{th} = \frac{T_{SLA}}{N_{IP}} \sum_{i=1}^{N_{IP}} \sum_{j=1}^{N_T(IP_i)-1} \frac{MSD_{j,j+1}(IP_i)}{N_T(IP_i)} \dots (3.2)$$

Where, N_{IP} represents number of unique IP addresses from which tasks are input for processing & scheduling purposes, while T_{SLA} represents Service Level Agreement timing threshold, which is set by cloud designers.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
DOI: https://doi.org/10.17762/ijritcc.v11i9.8114
Article Received: *07 July 2023* ***Revised:*** *28 August 2023* ***Accepted:*** *16 September 2023*

Tasks with $MSD > MSD_{th}$ are filtered, and the $(i+1)^{th}$ task is shifted to the end of scheduling queue, which assists in maintaining task-SLAs. A task chronology metric is evaluated for each set of tasks via equation 3.3,

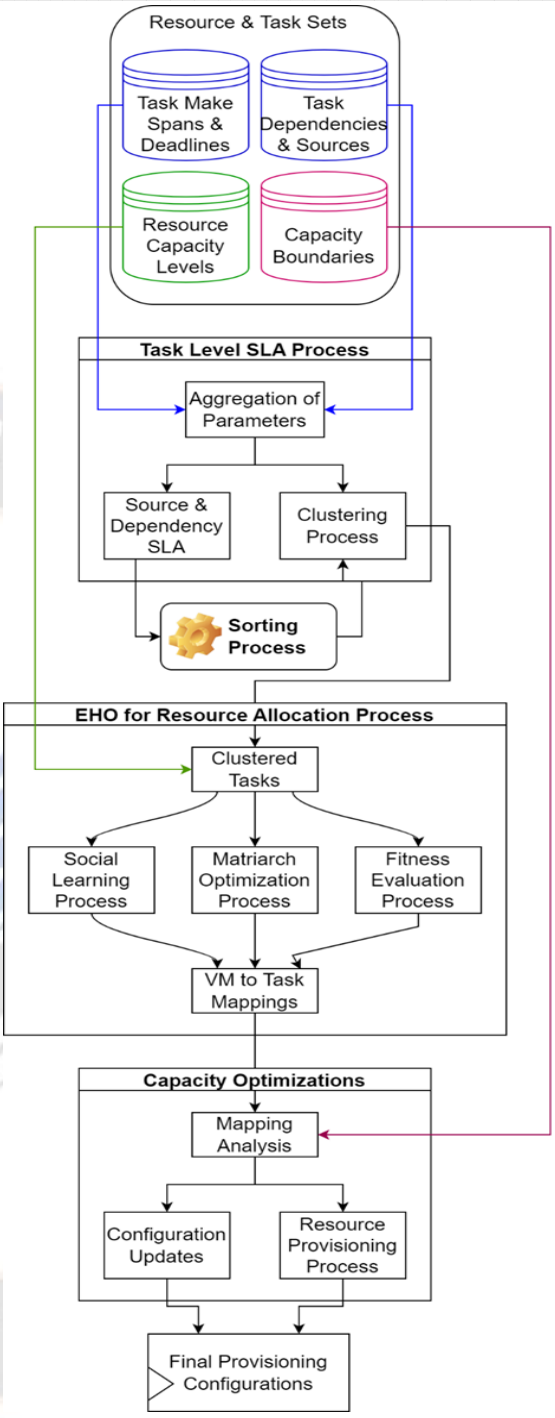
$$TCM_{i,j} = D(i,j) - D(j,i) \dots (3.3)$$

Where, TCM represents task chronology metric, while $D(i,j)$ represents dependency of task i on task j, which is a binary flag provided by input task sets. If $TCM \geq 0$, then the task sequence is unaltered, else tasks are swapped in order to maintain dependency correctness.

These tasks are clustered via different clustering processes, which assists in grouping them into make-span and deadline aware clusters. After the enforcement of SLA, initial task sequences are established to facilitate the execution of incoming tasks from diverse users and machines. However, these tasks are not bound by any time limits or resource constraints, which can impede their effectiveness. In this section, a suggested ensemble pattern analysis model is presented as a solution to tackle this problem. The model aims toimprove the execution efficiency of tasks by grouping those with comparable performance characteristics. In order to accomplish this goal, equation 4 is utilized to evaluate a novel task rank metric (TRM).

$$TRM_i = \frac{T_{p_i}}{N} * \left[\frac{Len_{T_i}}{Max(Len_{T_i})} + \frac{BW_{T_i}}{Max(BW_{T_i})} + \frac{RAM_{T_i}}{Max(RAM_{T_i})}\right] * \frac{DL_{T_j}}{\sum_{j=1}^{N} DL_{T_j}} \dots (4)$$

Where, T_p is position of the task as decided by the SLA model, Len represents make-span of the task, BW represents bandwidth needed for executing the task, RAM represents RAM needed for task execution, and DL represents task deadlines, while N indicates the number of tasks that are in pipeline for execution process. The TR metric of each task is inputted into three different models: hierarchical clustering, kMeans clustering, and FCM. The outcomes of each model are assessed, and to determine the ideal number of clusters, Equation 5 is utilized.



$$CN_{final} = CN_{HC}, when\ [CN_{HC} = CN_k = CN_{FCM}\ or\ CN_{HC} = CN_k\ or\ CN_{HC} = CN_{FCM}],$$
$$else, CN_{final} = CN_k, when\ [CN_k = CN_{HC}\ or\ CN_k = CN_{FCM}]$$
$$else, CN_{final} = CN_{FCM} \dots (5)$$

Where, CN represents cluster number, which assists in estimation of intra-cluster metric via equation 6,

$$ICM_i = \sqrt{\frac{\sum_{i=1}^{m}(TRM_i - \frac{\sum_{i=1}^{m}\sqrt{\frac{\sum_{j=1}^{n}(TRM_j - \frac{\sum_{k=1}^{n} TRM_k}{n})^2}{n-1}}}{m})^2}{m-1}} \dots (6)$$

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
DOI: https://doi.org/10.17762/ijritcc.v11i9.8114
***Article Received:** 07 July 2023 **Revised:** 28 August 2023 **Accepted:** 16 September 2023*

Where, m represents total tasks present in the current cluster set, while n represents total tasks present in other clusters. The intra cluster metric (ICM) values are arranged in descending order, which represents higher to lower computation task sets. These task sets are assigned to respective resources via a EHO based process.scheduling, the upcoming section of the text provides an explanation for this.

B.EHO Model design for VM to Task Mapping process

The SLA based model assists in maintaining dependency awareness along with task-level SLA with relation to various task types. An optimization procedure using EHO is used to process these rearranged tasks, that takes into consideration task deadline and make-span levels. This model works via the following process,

- Initially, optimization parameters of EHO are initialized as follows,
  - Total size of the optimization Herd (H_{size})
  - The total number of herds utilised for optimisation (N_{herds})
  - Total number of iterations used for optimization (N_i)
  - Learning speed of EHO Model (L_r)
  - Total resources or VMs to be processed for each herd (N_{hm})
- For each resource, initialize the following parameters,
  - The resource's capability is denoted in terms of millions of instructions per second (MIPS or M).
  - Total processing units (C)
  - Available RAM Memory (R)
  - Available Bandwidth with the resource (B)
  - Total capacity of computation for each resource (CC)
- Scan each herd for all iterations between 1 and N_i, perform following process,
  - Initialize a stochastic value of learning rate L_r
  - Map the tasks in sequence with stochastic resources
  - Calculate herd fitness after stochastic allocation via following equation 7,

$$f_i = \frac{\sum S_{task}}{\sum CC} + \frac{\sum BW_{task}}{\sum B} + \frac{\sum RAM_{task}}{\sum R} + \frac{\sum DL_{task}}{\sum C * \frac{R}{B}} \dots (7)$$

- After an iteration is completed, evaluate fitness threshold via equation 8,

$$f_{th} = \sum_{i=1}^{N_{herds}} \frac{f_i * L_r}{N_{herds}} \dots (8)$$

- Modify herds with $f \geq f_{th}$ in the next iteration, while use other herds directly for fitness evaluation in consecutive iterations.

Due to inclusion of make-span and task deadline, the process assists in mapping resources to tasks with better efficiency levels. But in some cases, VMs are either not used to their full capacity, or are over burdened with tasks. To avoid such scenarios, a novel Q-Learning model for VM Provisioning via capacity tuning process is deployed, and discussed in the next section of this text.

C. Integration of Q-Learning for VM Provisioning via capacity tuning process

A Q-Learning based VM Provisioning model is employed to enhance resource allocation efficiency, by iteratively optimizing processing parameters. The process followed to achieve this objective is as follows,

- Let initial capacity for each resource be IC, which is evaluated via equation 9,

$$IC = \sum_{i=1}^{C} CC_i * RAM_i * MIPS_i \dots (9)$$

- This is evaluated for each resource (or VM), and VM tasks are assigned depending on the EHO process.
- Once the current task-batch is executed, then VM utilization factor (VUF) is evaluated via equation 10,

$$VUF = \sum_{i=1}^{N_{vm}} N_{c_i} \dots (10)$$

Where, N_{vm} represents number of VMs (or resources) that participated in the resource allocation process, while N_c represents number of cycles for which each of these VMs were used to execute the tasks which were assigned to them via the EHO process.

- Now, increment capacity levels for each of these VMs via equation 11,

$$C(New) = C(Old)\left(\frac{IC + Max(IC)}{Max(IC)}\right) \dots (11)$$

Where, $C(New)$& $C(Old)$ represents new & old configuration parameters for each of the VMs, and $C \in (RAM, CC, MIPS)$

- Using this new configuration process next task-batch, and estimate new VM utilization factor via equation 10, which

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
DOI: https://doi.org/10.17762/ijritcc.v11i9.8114
***Article Received:** 07 July 2023 **Revised:** 28 August 2023 **Accepted:** 16 September 2023*

assists in identification of VM utilization for current task-set with current configuration of VMs.

- Now evaluate reward value via equation 12,

$$r = \frac{VUF(New) - VUF(Old) * (1 + L_r)}{L_r} - d * Max(VUF) \dots (12)$$

Where, d represents a discount factor, which is evaluated via equation 13,

$$d = \frac{N_{vm}(Old)}{N_{vm}(New)} \dots (13)$$

- If $r > 0$, that indicates that current configuration of VM is not sufficient, and it must be incremented via equation 11, else if $r < 0$, then configuration of VM is reduced via equation 14,

$$C(New) = C(Old)\left(\frac{Max(IC) - IC}{Max(IC)}\right) \dots (14)$$

- This process is repeated for each task-set, until $r \neq 0$

As a result of this ongoing optimization, the model can effectively adjust the capacity of VMs to improve their scheduling abilities. These capabilities are evaluated according to deadline hit ratio, make-span,computational cycles, energy consumption, and computational complexity, in this research work's subsequent section, Other models also experienced this evaluation also compared with the proposed model for validation purposes.

IV. RESULTS AND COMPARISION

According to the description of the proposed model, a combination of EHO with Q-Learning and SLA enforcement is capable of including deadline awareness, SLA rules, and incremental optimization under dependent task types. Verifying the model's performance, it was assessed based on its task make-span delay (D), computational cycles needed (CC), deadline hit ratio (DHR), and energy efficiency (E) under various VM and task settings. This performance was evaluated in comparison to DNNS [9], NPS CS [13], and EES [14]. To do this comparison, the Parallel Workloads Archive (PWA), which can be seen at https://www.cs.huji.ac.il/labs/parallel/workload [32], was used to retrieve configurations for various VM & task types and used with open-source licences. Sandia Ross cluster logs, San Diego Supercomputer Centre (SDSC) Blue Horizon logs, Lawrence Livermore National Lab's Linux Cluster Logs, Potsdam Institute for Climate Impact Research (PIK) IBM iDataPlex Cluster logs, and Intel Netbatch logs were used for the assessment process.

The proposed model, NASLMRP was validated using the Cloud Sim simulator, which aided in the formation of virtual machines (VMs or resources), job scheduling procedures, and assessment of the performance of various model parameters. These datasets were pooled to provide 1 million jobs, which were tested on 400 virtual machines with standard setups. Based on this assessment technique, the average make-span delay (D) for various Number of Tasks (NT) can be observed from Table 1 as follows,

Table 1. Make-span for different models under multiple task types

NT	D (ms) DNNS [9]	D (ms) NPS CS [13]	D(ms) EES [14]	D (ms) NAS LMRP (Proposed model)
10k	0.12	0.13	0.15	0.05
20k	0.14	0.15	0.18	0.06
30k	0.16	0.17	0.21	0.07
40k	0.18	0.21	0.25	0.09
50k	0.21	0.26	0.3	0.1
60k	0.24	0.32	0.36	0.12
70k	0.29	0.4	0.44	0.15
80k	0.34	0.49	0.53	0.18
90k	0.42	0.6	0.65	0.22
100k	0.52	0.73	0.79	0.27
200k	0.63	0.87	0.94	0.32
300k	0.76	1.02	1.1	0.38
400k	0.89	1.18	1.23	0.43
500k	1	1.31	1.33	0.48
700k	1.13	1.5	1.42	0.53
800k	1.17	1.56	1.48	0.55
900k	1.21	1.62	1.55	0.57
1M	1.26	1.69	1.64	0.6

After evaluating the proposed model and referring to figure 3, it can be concluded that the model exhibits a performance improvement of 38.5% when compared to DNNS[9] , 46.1% faster performance than NPS CS[13] , and 45.5% faster performance than EES[14], which makes it useful for low-delay scheduling scenarios. As a result of incorporation of resource capacity and task make-span levels along with incremental performance tuning, which assists in identification of optimal scheduling configurations

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
DOI: https://doi.org/10.17762/ijritcc.v11i9.8114
Article Received: *07 July 2023* ***Revised:*** *28 August 2023* ***Accepted:*** *16 September 2023*

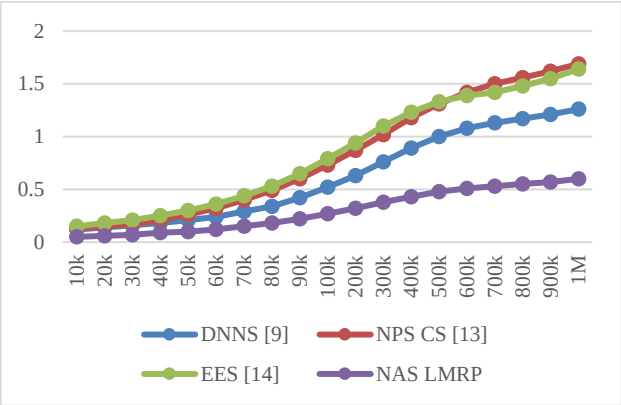


Figure 3. Make-span for different models under multiple task types

To further estimate model's performance, its Deadline Hit Ratio (DHR) was evaluated via figure 3

$$DHR = \frac{N_{t_d}}{T_t} \dots (15)$$

Where, N_{t_d} represents total tasks that were executed before their deadlines, and T_t represents count of total tasks which were given to the cloud for scheduling purposes. This evaluation resulted in the calculation of DHR in relation to the count of tasks executed in total is provided in Table 2 displayed below.

Table 2. DHR levels for different models under multiple task types

NT	DHR (%) DNNS[9]	DHR(%) NPS CS[13]	DHR(%) EES[14]	DHR (%) NASLMRP (Proposed model)
10k	93.65	93.67	93.44	99.16
20k	93.67	93.7	93.46	99.16
30k	93.7	93.72	93.49	99.17
40k	93.72	93.75	93.52	99.18
50k	93.75	93.78	93.55	99.18
60k	93.78	93.81	93.59	99.18
70k	93.81	93.84	93.61	99.19
80k	93.84	93.87	93.64	99.19
90k	93.87	93.9	93.67	99.2
100k	93.9	93.93	93.7	99.21
200k	93.93	93.96	93.72	99.22
300k	93.96	93.99	93.75	99.23
400k	93.99	94.01	93.78	99.24
500k	94.01	94.04	93.81	99.25
600k	94.04	94.07	93.85	99.25
700k	94.07	94.09	93.89	99.26
800k	94.09	94.12	93.92	99.27
900k	94.12	94.15	93.94	99.27
1M	94.15	94.18	93.97	99.28

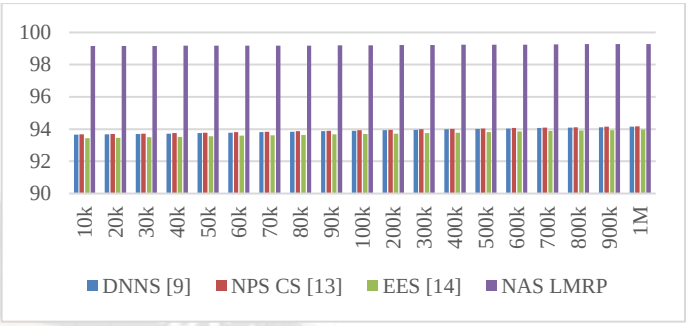


Figure 4. DHR levels for different models under multiple task types

The evaluation results depicted in Figure 4, demonstrate that the proposed model exhibits a 4.8% higher DHR than DNNS[9], 4.5% higher DHR than NPS CS[13] , and 5.9% higher DHR than EES[14], which makes it useful for deadline aware scheduling considerations. This results from being included of resource capacity with task deadlines and incremental performance tuning, which assists in identification of optimal schedulingconfigurations. Similarly, the computational cycles (CC) needed for execution is evaluated via equation 16, and represents the efficiency with which tasks are executed on different resource types.

$$E = \frac{NCC_{opt}}{NCC} * NT \dots (16)$$

Where, NCC_{opt} represents ideal cycles which must be used for execution of tasks, while NCC represents actual number of cycles needed for execution of these tasks. Based on this evaluation, computational cycles (CC) needed for execution is tabulated in table 3. as follows,

Table 3 Computational Cycles needed for execution of different task types

NT	CC DNNS [9]	CC NPS CS [13]	CC EES [14]	CC NAS LMRP (Proposed model)
10k	152.59	157.04	155.28	141.87
20k	153.8	157.66	156.19	142.76
30k	155	158.28	157.1	143.65
40k	156.2	158.91	158.01	144.53
50k	157.38	159.53	158.91	145.4
60k	158.56	160.16	159.81	146.28
70k	159.75	160.78	160.72	147.16

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
DOI: https://doi.org/10.17762/ijritcc.v11i9.8114
***Article Received:** 07 July 2023 **Revised:** 28 August 2023 **Accepted:** 16 September 2023*

80k	160.94	161.41	161.63	148.04
90k	162.13	162.03	162.54	148.92
100k	163.32	162.66	163.44	149.79
200k	164.51	163.28	164.35	150.67
300k	165.7	163.91	165.26	151.55
400k	166.89	164.53	166.16	152.43
500k	168.08	165.15	167.07	153.31
600k	169.27	165.78	167.98	154.19
700k	170.46	166.4	168.88	155.07
800k	171.65	167.03	169.79	155.95
900k	172.84	167.65	170.7	156.83
1M	174.04	168.27	171.6	157.71

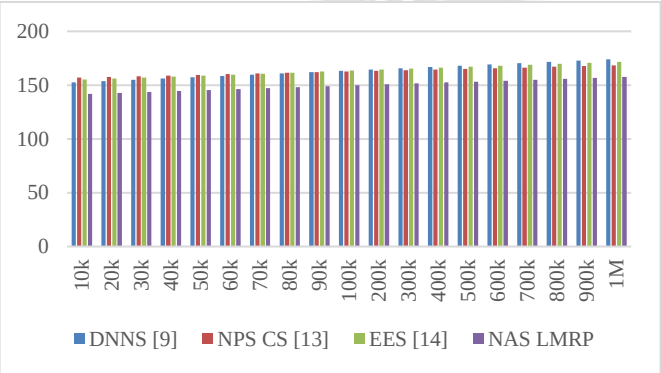


Figure 5: Computational Cycles needed for execution of different task types

Upon analyzing the outcomes and referencing figure 5, it becomes evident that the proposed model consumes 15% fewer computational cycles in comparison to DNNS[9], 8.3% lower computational cycles than NPS CS[13] and 12.5% lower computational cycles than EES[14] , which makes it useful for computationally efficient scheduling situations. This is because it incorporates resource capacity with incremental performance tuning, which assists in identification of optimal schedulingconfigurations. Similarly, the energy required for execution of different task types is tabulated in table 4 as follows,

Table 4. Energy required by different models for the scheduling process

NT	E (mJ) DNNS[9]	E (mJ) NPS CS [13]	E (mJ) EES [14]	E (mJ) NASLMRP (Proposed model)
10k	312.81	279.94	187.24	172.95
20k	315.29	281.05	188.34	174.08
30k	317.76	282.16	189.44	175.22
40k	320.2	283.28	190.53	176.33
50k	322.62	284.39	191.62	177.43
60k	325.04	285.5	192.72	178.54
70k	327.48	286.61	193.81	179.65
80k	329.92	287.73	194.9	180.76
90k	332.36	288.84	196	181.88
100k	334.8	289.95	197.09	182.99
200k	337.24	291.07	198.18	184.1
300k	339.68	292.18	199.28	185.22
400k	342.12	293.29	200.37	186.33
500k	344.56	294.4	201.47	187.44
600k	347	295.51	202.56	188.56
700k	349.44	296.63	203.66	189.67
800k	351.88	297.74	204.75	190.78
900k	354.33	298.85	205.84	191.9
1M	356.77	299.97	206.94	193.01

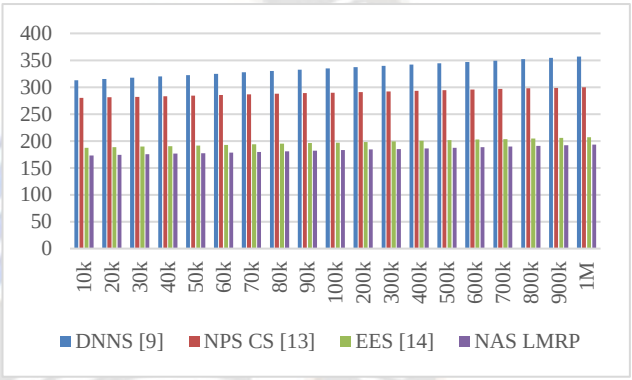


Figure 6. Energy required by different models for the scheduling process

Based on this assessment and figure 6, it can be seen that the proposed model uses 24.5%, 18.3%, and 16.5% less computational energy than DNNS [9], NPS CS [13], and EES [14], respectively, making it suitable for low energy scheduling scenarios. This is because incremental performance tuning combined with the incorporation of resource capacity, RAM, MIPS, and the number of VMs aids in the identification of the best scheduling configurations. The proposed model is useful for large-scale task scheduling application use cases as a result of these improvements.

V. CONCLUSION AND FUTURE WORK

The proposed model is able to fuse SLA, multiple level clustering, EHO based task scheduling, and Q-Learning based resource capacity optimizations in order to improve scheduling performance under heterogeneous & dependent task types. This performance was evaluated in terms of different quantitative metrics, and was compared with various state-of-the-art methods. This comparison revealed that the proposed

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
DOI: https://doi.org/10.17762/ijritcc.v11i9.8114
***Article Received:** 07 July 2023 **Revised:** 28 August 2023 **Accepted:** 16 September 2023*

model performs 38.5% faster than DNNS [9], 46.1% faster than NPS CS [13], and 45.5% faster than EES [14], making it useful for low-delay scheduling scenarios. This is because incorporating task make-span and resource capacity levels, along with incremental performance tuning, helps identify the best scheduling configurations. The proposed model's higher deadline hit ratios—4.8% higher DHR than DNNS [9], 4.5% higher DHR than NPS CS [13], and 5.9% higher DHR than EES [14]—make it useful for scenarios involving deadline-conscious scheduling. This is because task deadlines and resource capacity are combined, and incremental performance tuning helps identify the best scheduling configurations. Although it can also be seen that the proposed model needs 15%, 8.3%, and 12.5% fewer computational cycles than DNNS [9], NPS CS [13], and EES [14], respectively, which makes it useful for computationally efficient scheduling scenarios. This is because incremental performance tuning and resource capacity integration help identify the best scheduling configurations. While it was discovered that the proposed model, when compared to DNNS [9], NPS CS [13], and EES [14], uses 24.5%, 18.3%, and 16.5% less computational energy, respectively, making it appropriate for low energy scheduling scenarios. This is because identifying the optimal scheduling configurations is made easier by combining incremental performance tuning with the incorporation of resource capacity, RAM, MIPS, and the number of VMs. These improvements make the proposed model suitable for large-scale task scheduling application use cases.

In future, researchers can extend performance of the proposed model via integration of multiple deep learning models including Convolutional Neural Networks, Gated Recurrent Units, Auto Encoders, and Generative Adversarial Networks, which will assist in task pre-emption for optimization of scheduling pre-emption capabilities. Researchers can also integrate different bioinspired models including Grey Wolf Optimization (GWO), Firefly Optimization (FFO), etc. which will assist in optimizing & stabilizing performance of the proposed model under different real-time task types.

REFERENCES

[1] Mahmoud, H., Thabet, M., Khafagy, M. H., & Omara, F. A., Multiobjective task scheduling in cloud environment using decision tree algorithm. IEEE Access, 10, 36140-36151,2022

[2] Chai, R., Li, M., Yang, T., & Chen, Q. (2021). Dynamic priority-based computation scheduling and offloading for interdependent tasks: leveraging parallel transmission and execution. IEEE Transactions on Vehicular Technology, 70(10), 10970-10985.,2021

[3] Alsadie, D., A metaheuristic framework for dynamic virtual machine allocation with optimized task scheduling in cloud data centers. IEEE Access, 9, 74218-74233,2021

[4] Lee, Y. S., & Han, T. H., Task parallelism-aware deep neural network scheduling on multiple hybrid memory cube-based processing-in-memory. IEEE Access, 9, 68561-68572.,2021

[5] 11.Mao, R., & Aggarwal, V., NPSCS: Non-preemptive stochastic coflow scheduling with time-indexed LP relaxation. IEEE Transactions on Network and Service Management, 18(2), 2377-2387.,2021

[6] Aejaz Farooq Ganai, Farida Khursheed. (2023). Computationally Efficient Holistic Approach for Handwritten Urdu Recognition using LRCN Model. International Journal of Intelligent Systems and Applications in Engineering, 11(4s), 536 –. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/2724

[7] Zhang, Q., Gui, L., Zhu, S., & Lang, X. ,Task offloading and resource scheduling in hybrid edge-cloud networks. IEEE Access, 9, 85350-85366.,2021

[8] Chen, X., Cheng, L., Liu, C., Liu, Q., Liu, J., Mao, Y., & Murphy, J. , A WOA-based optimization approach for task scheduling in cloud computing systems. IEEE Systems journal, 14(3), 3117-3128.,2020

[9] Gammoudi, A., Benzina, A., Khalgui, M., & Chillet, D., Energy-efficient scheduling of real-time tasks in reconfigurable homogeneous multicore platforms. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 50(12), 5092-5105.,2018

[10] Yuan, H., Bi, J., & Zhou, M., Profit-sensitive spatial scheduling of multi-application tasks in distributed green clouds. IEEE Transactions on Automation Science and Engineering, 17(3), 1097-1106,2019

[11] Qi, Q., Zhang, L., Wang, J., Sun, H., Zhuang, Z., Liao, J., & Yu, F. R. Scalable parallel task scheduling for autonomous driving using multi-task deep reinforcement learning. IEEE Transactions on Vehicular Technology, 69(11), 13861-13874, 2020.

[12] Jiang, E., Wang, L., & Wang, J., Decomposition-based multi-objective optimization for energy-aware distributed hybrid flow shop scheduling with multiprocessor tasks. Tsinghua Science and Technology, 26(5), 646-663.,2021

[13] Kessler, C., Litzinger, S., & Keller, J., Static scheduling of moldable streaming tasks with task fusion for parallel systems with DVFS. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 39(11), 4166-4178,2020

[14] Chen, L., Zhu, J., Deng, Y., Li, Z., Chen, J., Jiang, X., & Liu, L., An Elastic Task Scheduling Scheme on Coarse-Grained Reconfigurable Architectures. IEEE Transactions on Parallel and Distributed Systems, 32(12), 3066-3080.,2021

[15] Nabi, S., Ibrahim, M., & Jimenez, J. M., DRALBA: Dynamic and resource aware load balanced scheduling approach for cloud computing. IEEE Access, 9, 61283-61297,2021

[16] Marahatta, A., Pirbhulal, S., Zhang, F., Parizi, R. M., Choo, K. K. R., & Liu, Z., Classification-based and energy-efficient dynamic task scheduling scheme for virtualized cloud data center. IEEE Transactions on Cloud Computing, 9(4), 1376-1390,2019.

[17] Chen, Z., Hu, J., Chen, X., Hu, J., Zheng, X., & Min, G., Computation offloading and task scheduling for DNN-based applications in cloud-edge computing. IEEE Access, 8, 115537-1155472,2020

[18] Chaudhary, D. S. ., & Sivakumar, D. S. A. . (2022). Detection Of Postpartum Hemorrhaged Using Fuzzy Deep Learning Architecture . Research Journal of Computer Systems and Engineering, 3(1), 29–34. Retrieved from https://technicaljournals.org/RJCSE/index.php/journal/article/view/38

[19] Ali, A., & Iqbal, M. M., A cost and energy efficient task scheduling technique to offload microservices based applications in mobile cloud computing. IEEE Access, 10, 46633-46651,2022

[20] Quan, Z., Wang, Z. J., Ye, T., & Guo, S, Task scheduling for energy consumption constrained parallel applications on heterogeneous computing systems. IEEE Transactions on Parallel and Distributed Systems, 31(5), 1165-1182,2019.

[21] Yuan, H., Tang, G., Li, X., Guo, D., Luo, L., & Luo, X. , Online dispatching and fair scheduling of edge computing tasks: A learning-based approach. IEEE Internet of Things Journal, 8(19), 14985-14998,2021.

[22] Jiang, X., Sun, J., Tang, Y., & Guan, N., Utilization-tensity bound for real-time DAG tasks under global EDF scheduling. IEEE Transactions on Computers, 69(1), 39-50.2019

[23] Orr, M., & Sinnen, O., Integrating task duplication in optimal task scheduling with communication delays. IEEE Transactions on Parallel and Distributed Systems, 31(10), 2277-2288, 2020

[24] Wang, Y., & Zuo, X., An effective cloud workflow scheduling approach combining PSO and idle time slot-aware rules. IEEE/CAA journal of automatica sinica, 8(5), 1079-1094.2021

[25] Mrs. Ritika Dhabliya. (2020). Obstacle Detection and Text Recognition for Visually Impaired Person Based on Raspberry Pi. International Journal of New Practices in Management and Engineering, 9(02), 01 - 07. https://doi.org/10.17762/ijnpme.v9i02.83

[26] Rajesh, M., Analysis and Design of Advance Scalable QoS Based Resource Provisioning Framework. Recent Trends in Intensive Computing, 39, 114. 2021

[27] Shelke, P., & Shahapurkar, R., TS2LBDP: Design of an Improved Task-Side SLA Model for Efficient Task Scheduling via Bioinspired Deadline-Aware Pattern Analysis. International Journal of Intelligent Information Technologies (IJIIT), 18(3), 1-13.,2022

[28] Shelke, P., & Shahapurkar, R., Analysis of time factor with resource provisioning frameworks in a cloud environment for improving scheduling performance. International Journal of Next-Generation Computing, 13(3),2022

[29] Feitelson, D. G., Tsafrir, D., & Krakov, D., Experience with using the parallel workloads archive. Journal of Parallel and Distributed Computing, 74(10), 2967-2982,2014