Journal of Ambient Intelligence and Humanized Computing (2022) 13:1153–1171
https://doi.org/10.1007/s12652-021-03537-7

ORIGINAL RESEARCH

A planned scheduling process of cloud computing by an effective job allocation and fault-tolerant mechanism

Manoj Kumar Malik[1] · **Ajit Singh**[2] · **Abhishek Swaroop**[3]

Received: 8 October 2020 / Accepted: 7 October 2021 / Published online: 8 January 2022

Abstract

In the scientific world, cloud computing is utilized for several applications like financial, healthcare biomedical systems, and so on. However, the chief drawback behind in cloud computing paradigm is if any one of the hosts failed during the data transmission then it interrupts the whole process. To overcome this problem the current research proposed a novel Hybrid Grey Wolf and Ant Lion Model (HGW–ALM) with lively standby replication (LSR) strategy to enhance the cloud computing paradigm. In addition, if any one of the hosts has less capacity than its workload, then that particular host is predicted by the HGW–ALM model and the specified host is maintained by the LSR approach. Moreover, the checkpoint strategy is efficiently processed with the tolerant mechanism. In addition, the discussed faults in this present article were virtual machine failure faults, timing faults, and response faults. Also, the robustness of the proposed algorithm is checked against few attacks like replay, Denial of service and data injection attacks. Subsequently, the drawn charts, graphs, and tables proved the efficiency of the proposed work by comparing key metrics with existing approaches. Thus, the proposed frame model achieved a better result by obtaining high throughput as 6000 bps, resource usage only 20% and less makespan time 200 s.

Keywords Job scheduling · Fault tolerant in cloud computing · Load balancing · Execution time · Cloud computing

Abbreviations

VM	Virtual machine
LSR	Lively standby replication
HGW–ALM	Hybrid Grey Wolf and Ant Lion Model
DoS	Denial of service
MER	Maximum effective reduction
HGPSO	Hybrid genetic-particle swarm optimization
ABC	African Bee Colony
SSO	Shark smell optimization
IGDT	Information gap decision theory
CRCH	Checkpointing, and replication based on clustering heuristics
HEFT	Heterogeneous earliest finish time
ABC	Artificial bee colony
Bi-LSTM	Bi-directional long short term memory
QoS	Quality of service
AL	Ant Lion
GW	Grey Wolf
MIPS	Multi instructions per second

✉ Manoj Kumar Malik
manojmalik@msit.in

Ajit Singh
erajit@rediffmail.com

Abhishek Swaroop
asa1971@gmail.com

[1] Department of Information Technology, Maharaja Surajmal Institute of Technology, Janak Puri, New Delhi 110058, India

[2] Department of Computer Science and Engineering, Bipin Tripathi Kumaon Institute of Technology, Dwarahat, Uttarakhand 263653, India

[3] Department of Information Technology, Bhagwan Parshuram Institute of Technology, Rohini, New Delhi 110089, India

1 Introduction

The minimized costs of storage and processing technologies made the sudden growth of computing resources manufacturing (El Makkaoui et al. 2019). Recently, a novel computing model called cloud computing is developed to provide an on-demand network admittance to the network users through resource virtualization (Bhushan and Gupta 2019). In addition, cloud computing is utilized in many applications, so it needs a fault-tolerant model to provide uninterrupted communication (Ponmagal et al. 2020). Also,

an effective scheduling mechanism in cloud computing has reduced the scheduling and execution time (Mukherjee et al. 2020). Thus, the cloud scheduling model is applicable in different sectors and many fields such as scientific appliances, data mining, artificial intelligence, and the internet of things (Qiang 2019). Due to its economical pay per use model, cloud computing was considered the most suitable computing frame model for large computational applications (Park 2018). In a cloud processing environment the customers or users outsource their statistics to the cloud, it performs the calculating operations and saves the results (Latiff et al. 2018). Thus the consistency is one of the greatest concerns in cloud computing. In cloud computing to process the job execution, initially, the tasks are divided and arranged based upon the priority. Moreover, cloud computing has a number of virtual machines to execute the allocated jobs (Tamilvizhi and Parvathavarthini 2019).

In addition, crossing the deadline of each task is considered a timing fault; in that case, the server migrates the job to another virtual machine that has the capacity to complete the task within a short time (Ahmad et al. 2019). Also, during the job execution, if the files are large, then it takes more time period for an execution (Wu et al. 2019a,b). Hence, the processing time of the virtual machine (VM) (Li et al. 2020) is an important research problem that is being actively carried by the researchers (Yan et al. 2019). The key limitation behind in cloud computing environment is allocating the available jobs (Rodriguez and Buyya 2014). Therefore an excellent scheduling algorithm with a monitoring frame is required to allocate the jobs in an appropriate way to reduce the execution time and complexity (Zhu et al. 2016). Moreover, the main goal of cloud computing is preserving service to the users or customers with a substantial cost; also, it was preserved the services through various frameworks. The failure occurred, when there are any faults in arranged hubs (Yao et al. 2020), like high power consumption, less capacity, etc. Thus the fault occurrence in cloud infrastructure is an unpredictable manner (Ding et al. 2017). So, the fault-tolerant process is required to continue the process without any interruption (Sarmila et al. 2019). Furthermore, the chief issue in the cloud paradigm is faults occurring, so a fault-tolerant mechanism was introduced to tolerate the software faults such as response, timing, and transmission fault during the execution of the process. Hence, if there is checkpointing in the cloud data transaction, then it avoids data loss during VM faults by saving the copy of all data. So, in this case, the processed data during the fault was recovered successfully.

Over the years, several researchers have focused on fault-tolerant workflow scheduling to end this task scheduling issue (Suliman et al. 2019). Some of the proficient techniques are introduced such as replication mechanism (Lee and Gil 2019), integrated scheduling mechanism (Wu et al. 2019a,b), maximum effective reduction (MER) (Lee et al. 2015), etc., but still, the issues are not solved. Because, if the fault occurs in one connected VM, then it disturbs the whole system process; also, data loss occurs. Hence, the occurrence of a fault can terminate the whole process. The reason for raising fault is one VM can be capable of maintaining multi jobs at a single time. So if the job is overloaded then, VM has lost its capacity. To optimize this problem, some optimization models such as hybrid genetic-particle swarm optimization (HGPSO) (Kumar and Venkatesan 2019), African Bee Colony (ABC) (Thanka et al. 2019), etc., were implemented in past, but the successive results are not found. Hence, the main drawbacks behind those optimization strategies are the restricted job allocation. In those cases, if the VM has less capacity then it gets damages. However, cloud-based virtualization has attracted many researchers by its advancement.

So that, the hybrid optimization-based scheduling mechanism with a fault-tolerant strategy was proposed to enhance the job scheduling and a fault-tolerant mechanism of cloud computing. Also, the faults like VM failure faults, timing faults, and response faults were discussed in this present research. Also, the reliability of the system is analyzed by launching few attacks like wrong data injection attack (Liu et al. 2011), reply attack (Hosseinzadeh et al. 2019) and DoS attack (Bicakci and Tavli 2009). Moreover, different optimization algorithms are used in various applications for different purposes, such as stochastic optimization for load and cost prediction (Gao et al. 2019), general algebraic optimization for scheduling the power consumption of multi chiller (Saeedi et al. 2019), stochastic optimization for optimal renewable energy sharing (Abedinia et al. 2019), Pareto optimization for peak load management for industrial consumer (Khodaei et al. 2018), Shark smell optimization (SSO) for electricity load prediction (Ghadimi et al. 2018), information gap decision theory (IGDT) and robust optimization for risk assessment in photovoltaic wind battery (Bagal et al. 2018), etc. these reason has inspired this research towards executing the hybrid optimization in cloud paradigm for job scheduling and fault-tolerant process. Also, the chief highlight of this research article is the job migration strategy that can effectively improve the job allocation framework and has reduced job execution time. Also, the projected model is the broad framework that has the capacity to schedule and execute more tasks at a single time. The efficiency of the designed model is verified in the performance metrics comparison section. The key procedures of this research work are recapitulated as follows,

- Initially, several VM's are arranged for the job allocation and execution process in the java platform.
- Consequently, develop a novel Scheduling and detection of fault mechanism as HGW–ALM.

- The fitness function of the Grey Wolf model is utilized to assign the job based on the priority and the fitness of the Ant Lion model is used to find the fault in VM, such as VM failure faults, timing faults, and response faults.
- Once the fault is detected then immediately a novel Fault tolerant mechanism as LSR is developed.
- Also, the reliability of the system is analyzed by launching few malicious attacks like replay, fake data injection and DoS attacks.
- The proposed model is elaborated in java and its key metrics are validated with existing models and attained better results by achieving less execution time, makespan time and so on.

The remainder of this research is itemized as follows, Sect. 2 describes recent literature related to energy management in wireless sensor networks, Sect. 3 defines problem statement, Sect. 4 deals proposed methodology, Sect. 5 enumerate result and discussion and Sect. 6 concludes the paper.

2 Related work

Few recent studies related to fault-tolerant workflow scheduling in cloud computing are as follows.

A set of jobs are available in cloud computing; thus the workflow of cloud computing is planned in order to handle the jobs in a narrow way. So that Lee and Gil (2019) presented a checkpointing, and Replication based on Clustering Heuristics (CRCH) to schedule the job and tolerate the fault in the cloud framework. Moreover, the results of this proposed model prove the efficiency of scheduling and fault-tolerant strategies. Furthermore, the CRCH mechanism has three modules such as checkpointing, replication, and clustering. But in the evaluation, it takes more time to complete the process.

Nowadays, the usage of a multi-agent system is increased with great demand, so the efficient fault-tolerant mechanism in cloud computing is more important. Also, the cloud environment is gaining popularity among applications that required high computational obligations. So, Wu et al. (2019a,b) proposed an improved integrated scheduling mechanism for workflow scheduling and fault-tolerant process. Moreover, the proposed mechanism could stabilize the obtained execution results but it has utilized more resources to complete the assigned jobs.

Scientific workflows are often utilized in a cloud environment to validate the effective score of the utilized technique by analyzing its key parameters. Therefore, Setlur et al. (2019) have developed a Heterogeneous Earliest Finish Time (HEFT) with the synchronized lightweight model and checkpointing paradigm to reduce data loss during data broadcasting. But, it has taken more duration to execute the process because of checkpointing strategy.

Kumar and Venkatesan (2019) have developed a hybrid genetic-particle swarm optimization (HGPSO) model to optimize the scheduling tasks and time duration. Moreover, it process by separating all the job scheduling tasks into different levels based on the priority task order. Subsequently, it distributes the time limit to each and every level. All levels finished the task within the sub-time limit; finally, it has optimized the execution time. But, during the process, if any fault was raised, then the whole process gets disturbed.

The fault-tolerant mechanism is a key paradigm in cloud computing. So, Thanka et al. (2019) have been offered an enhanced Artificial Bee Colony (ABC) paradigm to schedule the task effectively in a priority manner. In addition, security measure is upgraded to protect the data from the third parties. Finally, by the validation, it was verified that the time cost was reduced considerably. One of the drawbacks of this model is it takes more time to execute the allocated jobs.

Ahmad et al. (2021) have developed awareness-based quality of service fault-tolerant work process management system to reduce the makespan time and resource cost. Finally, the designed framework is compared with other models, and its proficient score was estimated with the help of a scientific workflow model. In addition, 1000 tasks were taken to check the robustness and overhead of the projected model. By the validation, if the tasks are increased, then it takes more time to complete the process.

A wide range of computation power has been required to execute complex scientific tasks in cloud computing. Khaldi et al. (2020) have designed a clustering-based fault-tolerant mechanism for a real-time environment to check the competence of the developed model and maximum utilized power consumption. Finally, it has improved execution cost and makespan time. However, it has consumed more power.

Rezaeipanah et al. (2020) have planned to make the usual error detection framework in cloud computing. To process this function, initially, common errors are trained in the cloud environment. Consequently, fault analysis-based fuzzy logic was designed to detect the launched errors. Subsequently, the designed fuzzy model was applied in the cloud environment to validate the successive score of the designed model. By the validation, the designed model has diminished error occurrence and failure rate. But it takes more time to design the model. In another study, bidirectional long short term memory (Bi-LSTM) has proposed by Gao et al. (2020) to predict the failure in the cloud environment. At final, the Bi-LSTM model has reported 92% accuracy for predicting failure in the cloud data center, but it has consumed more energy.

Table 1 Related literatures comparison

References	Method	Metrics	Advantages	Demerits
Lee and Gil (2019)	CRCH	Accuracy: 99%, reduced execution time: 57%	High throughput ratio	Time complexity
Wu et al. (2019a,b)	Integrated scheduling mechanism	Algorithm rune time: 50, shrink ratio 0.3, execution unit 20	It has maintained the stability range	High power consumption
Setlur et al. (2019)	HEFT	Resource wastage reduction 46%, resource usage 35%	It has reduced the data flow rate	Time complexity
Kumar and Venkatesan (2019)	HGPSO	Completion time 700 ms, availability 92%	It has gained a high scheduling rate	In the presence of a fault, its performance is insufficient
Thanka et al. (2019)	Enhanced ABC	Time cost 210 s, execution time 160 s, task scheduling improvement 19.51%	It has improved the quality of service in cloud computing	It takes more time for execution
Ahmad et al. (2021)	Awareness-based QoS	Virtual machine 25–1000, Makespan 300 s, cost 7000 cents	It has gained a high reliability rate	It has required more time to complete the process
Khaldi et al. (2020)	Clustering-based fault-tolerant	execution time 4, makespan: 10,500 s, tasks 1000	Improved execution cost and reduced makespan time	High power consumption
Rezaeipanah et al. (2020)	Fuzzy model	Accuracy: 6.49%, makespan time 50 s, task: 200, response time: 0.15 s	It has minimized failure and error rate	Time complexity
Gao et al. (2020)	Bi-LSTM	Accuracy 92%, F1 score 86%	It has attained the finest failure prediction rate as 92% accuracy	It has consumed more energy

Hence the comparison of related works with merits and demerits are elaborated in Table 1.

3 System model and problem statement

Cloud computing is the set of large tasks which is allocated to each available VM in the computing series. The key drawback of cloud computing is an occurrence of a fault, the fault happens because of several reasons such as heavy load, link failure, a large computational task in a short execution time.

If one of the intermediate hosts gets damaged then it stopped the entire works. The system model of job allocation and execution is shown in Fig. 1. Many researchers have been proposed several approaches towards fault-tolerant workflow scheduling in a cloud environment. However, each of the proposed solutions has its own limitations and challenges. These issues motivated this work towards the area of fault-tolerant workflow scheduling in a cloud environment.

4 Proposed HGW–ALM with LSR

Usually, cloud computing is connected with a number of machines or devices for a particular process. The fault in the transmission link causes host failure; it tends to interrupt all other machines which are present in the series. So the proposed research aims to develop a novel HGWALM for the scheduling process and to detect different kinds of faults such as link failure faults, timing faults, and response faults. In addition, the development of the S-Guard Replication framework acts as a tolerant mechanism. Thus, the job is scheduled in an efficient manner using Grey Wolf (GW) fitness and a fault-tolerant mechanism is utilized to enhance the performance of the virtual machine. Moreover, if the server is failed then the work is shifted to another server by the load balancing strategy. The proposed work is shown in Fig. 2.

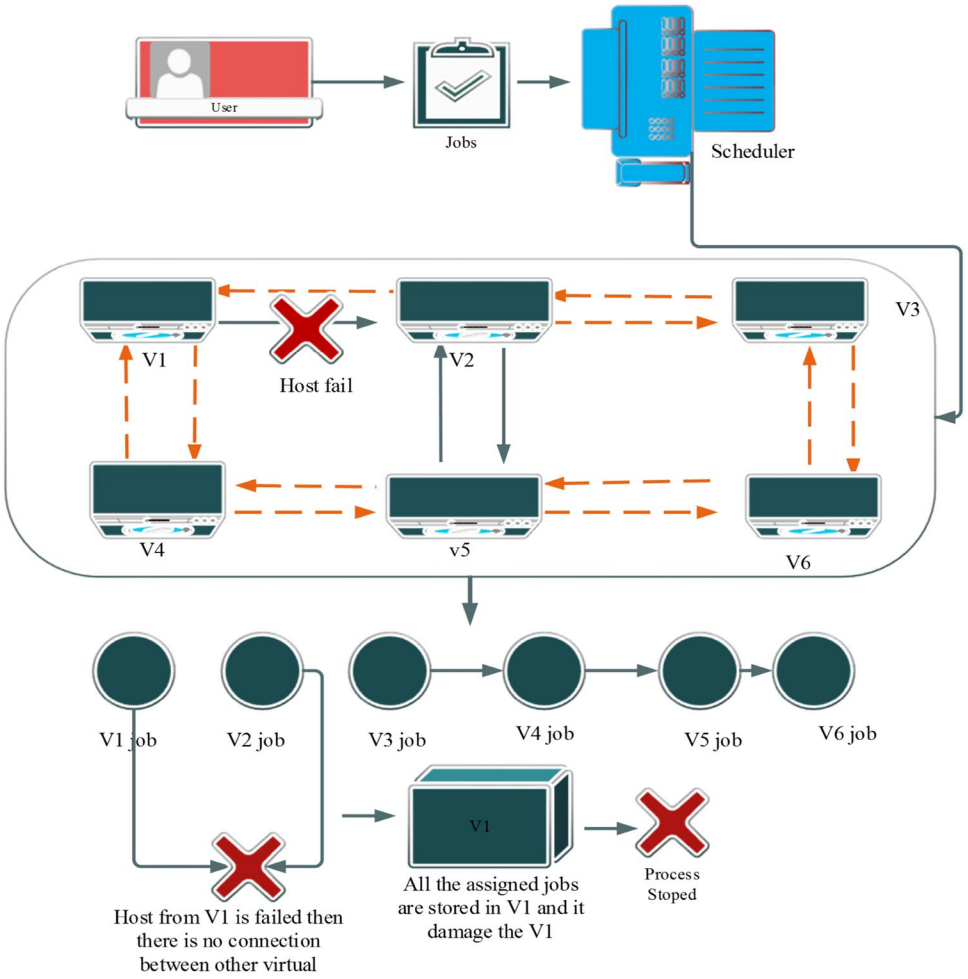

Fig. 1 System model with the problem in basic cloud-VM system

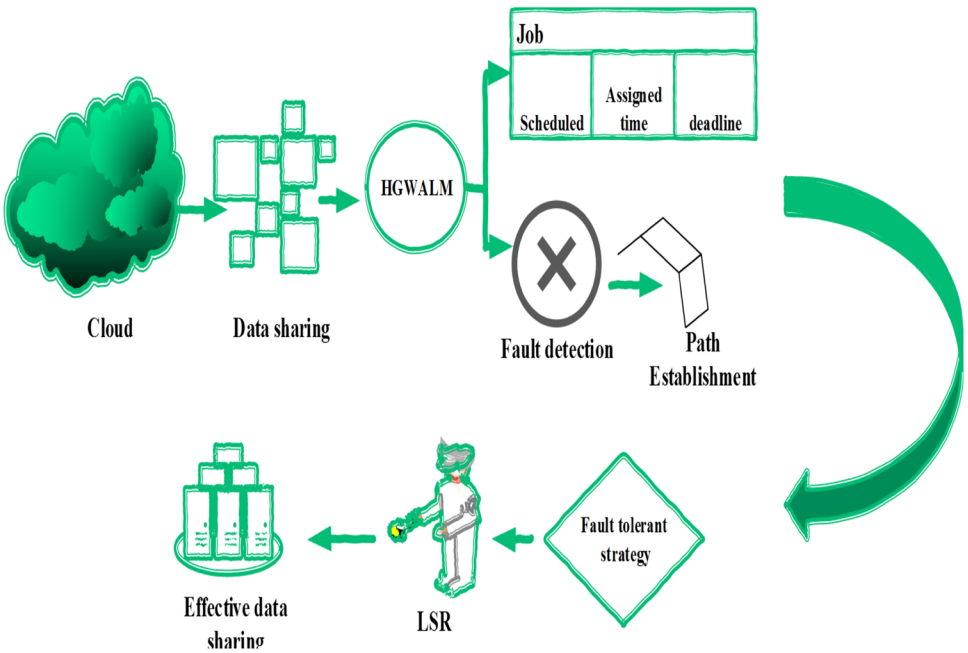

Fig. 2 Proposed HGW–ALM with LSR

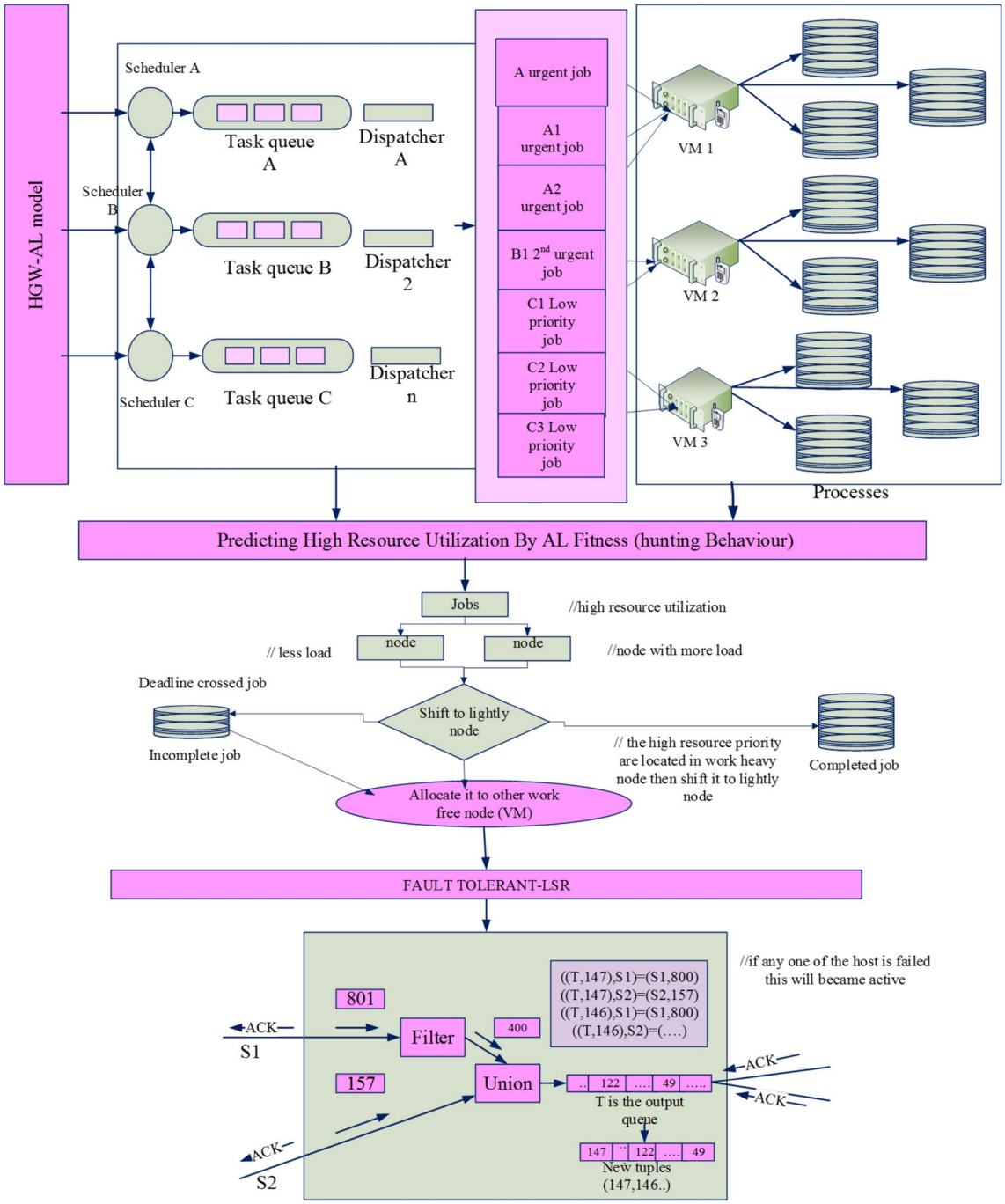

Fig. 3 Complete proposed flow model

Moreover, the faults VM are maintained by a novel fault-tolerant mechanism LSR. During the fault maintaining, the capacity of the node and its workload is checked if the workload is higher than the hub's capacity then that particular job is migrated to the other hubs with the Ant Lion (AL) fitness.

4.1 General working process of HGW–ALM with LSR

In cloud computing, workflow or job scheduling is one of the complicated tasks. Because, different computers, and operating systems have different capacities to satisfy the user

needs. Hence, an efficient scheduling strategy is required to use in the cloud environment. In this proposed scheme the job schedule mechanism should evaluate the following parameters such as, node energy, load balancing, task distribution, fault node monitoring, and approximate execution time which are detailed in Fig. 3.

Moreover, different user's requests are received by the cloud server for their basic needs and requirements. Furthermore, some task needs more computing time and resource utilization, also some other jobs need less computing time. So this research has focused on a heuristics algorithm to schedule the jobs and detect the faults during transmission. In addition, the work is assigned based on the priority which is to complete first; it's evaluated by the HGWAL mechanism. In cloud computing, some VM has less workloads and some VM's have more workloads, for that load balancing is one of the solution strategies in cloud computing. For the better allocation of resources, the outstanding strategy behind in cloud computing paradigm is load balancing. To enhance the system performance high resource estimation ratio is needed. The characteristics behind in load balancing approach is systemized as,

- Distributed work consistently across the hubs
- Enhancing the function of the system
- Minimize reply time
- To obtain high resource ratio utilization

If the communication delay is raised during the transmission, it may affect the balancing model. The complete flow of the work model is validated in Fig. 3.

4.1.1 A novel HGW–ALM scheduling model

In this current research work, Grey Wolf (Mirjalili et al. 2014) and Ant Lion (Mirjalili 2015) model is adopted to invent the new hybrid model. It is the arbitrary assignment model, here with the combination grey wolf and ant lion fitness, the hybrid optimization was developed for job scheduling and fault-tolerant process. In the initial phase, the grey wolf (Ahmad et al. 2021) fitness parameter is initialized as alpha (α), beta (β), delta (δ), and omega (ω). Here the (α) is to investigate all VMs and collect the status of node then the first priority based work is represented as (β) second priority job is represented as (δ) and the fault detected node is represented as ω. In the cloud environment job assigning model is detailed in the Eqs. (1) and (2). Here, the span time is represented as job completion time, hence, the assigned and completion time determining function is defined using Eqs. (1) and (2).

$$K_i^t = work_j^t + span\ time^t \tag{1}$$

$$K_i^t = work_j^t + assigned\ time^t \tag{2}$$

The VM represented as K, then i, j represents initialize and ending time, t denotes assigning time.

Algorithm 1: Pseudo-code for scheduling model HGW-ALM

Initialize *the number of nodes* $VM(i = 1,2,....n)$

Begin *with the jobs which have to be executed*

Estimate the fitness of all VM

(α) *monitor the cloud, and evaluate the conditions of all virtual machines*

if K *is the each individual VM, then* $K > 1$ *or* $|K| < -1$, *heavy workload VM*

Work deadline $\leq \alpha \leq executiontime$

Based on the priority, works are scheduled here, β **is the first priority job**

$R \geq \beta$ *//Beta checks resources for each work*

δ *third-best hunt operator (second priority job)*

while (t < total number of VM)

for *each monitoring agent* (α)

Once the first priority job is predicted by beta

end for

Update D *and* K *(update execution time, deadline and resource utilization*

End

Work deadline $t_i \max$,

establish the best while $(t_i > t_i max)$ ***work is completed***

Initiate iteration number $t_i = 2$

While $(t_{i_{crr}} < t_i max)$

for *every job* $(j = incompletework)$

for *every dimension* $(k = (lessworkVM)$

Initiate with uniform distribution by eqn. (6)

end for

for every $K(i = 0)$ *Fault is detected*

end for

Increment iteration $t_{icrr} = tolerant \quad mechanism$

end while

After the work is assigned to alpha it monitors the work schedule by its priority which is defined in Eq. (3). Here, E is the priority estimating parameter.

$$E_{\dot{\alpha}} = |D_1 \cdot Y_{\dot{\alpha}} - Y|, \quad E_{\dot{\beta}} = \left|D_2 \cdot Y_{\dot{\beta}} - Y\right|, \quad E_{\dot{\delta}} = |D_3 \cdot Y_{\dot{\delta}} - Y| \tag{3}$$

Immediately α collects the information of work deadline and first priority work is represented as β. Also, Y represents the deadline of each job and δ is the second priority job. In this procedure works set are determined as D = {D1, D2, D3,…,Dn} that workloads possessions in a cloud environment,

Here, the deadline of each job is represented as Y, before assigning a job, the capacity of VM should be evaluated using Eq. (4).

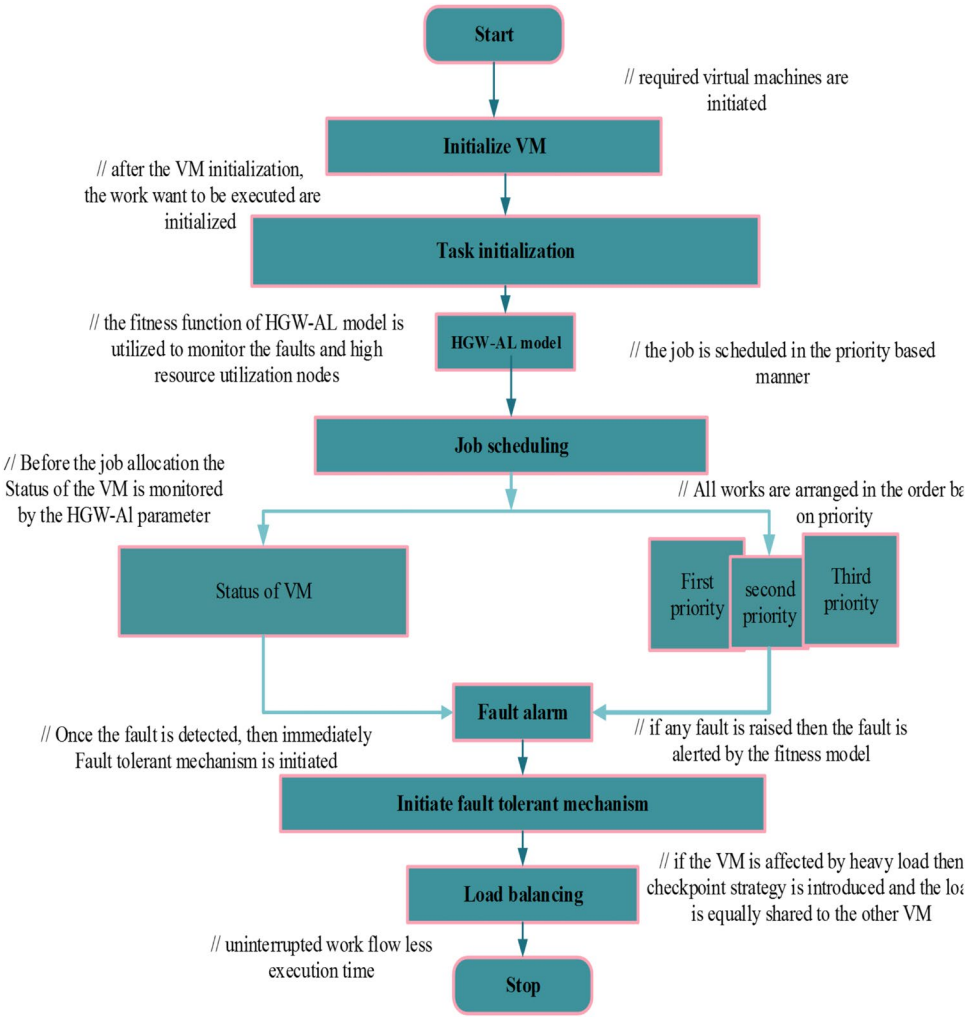

Fig. 4 Flow model of HGW-ALM with LSR

$$node_i^t = \frac{r_a + r_e}{2} \tag{4}$$

Here r_a represent the capacity of VM and r_e is the required resource to complete the task.

Here the work is scheduled under the priority-based model; once the alpha got all tasks of the work schedule then it monitors the network channel and collects the capacity of each VM. The capacity evaluation of each VM is detailed in Eq. (5).R is the available resource of each VM.

$$K_{\dot{\alpha}}^t = R_{\dot{\alpha}}^t + K^t, \quad K_{\dot{\beta}}^t = R_{\dot{\beta}}^t + K^t, \quad K_{\dot{\delta}}^t = R_{\dot{\delta}}^t + K^t \tag{5}$$

The resources set are considered $R = \{R1, R2, R3, R4, RD\}$ which is defined as input of the cloud; it is in the form of $D < R$. If the job and resources are assigned to each VM then the AL model initiates the fitness function by checking the deadline of each job. Moreover, if there is any deadline crossed work then the AL model transfers it to the VM which has sufficient resources to complete that task and it must be a lighter node. The lighter node selection is processed in Eq. (6).

$$K_\alpha = D_{\dot{\alpha}}^t + K_1^t, \quad K_{3\dot{\beta}} = D_{\dot{\beta}}^t + K^t, \quad K_{\dot{\delta}} = D_{\dot{\delta}}^t + K^t \tag{6}$$

While searching for a light VM to complete the task, if any one of the VM has $K = 0$ then it is a fault detected VM then immediately fault-tolerant process is initiated, which is illustrated in Algorithm 1 and Fig. 3.

In the cloud computing framework, the resources are offered based on their priority. Moreover, the priorities of various jobs are compared in an independent manner. Here, the normal work resource process is defined as δ and the priority-based significant work is arranged in a beta parameter.

Moreover, the task is assigned alpha continuously monitor the weight of nodes, if any one of the nodes drained its energy then the fault-tolerant mechanism replaced it with tuples to maintain the work without disturbance. The flow of work is modeled in Fig. 4.

4.2 Fault tolerant-LSR

After detecting the fault node using HGWALO that output is fed into a fault-tolerant mechanism subsequently the process

of fault-tolerant mechanism is processed. Moreover, the secondary user does not send any output data downstream. It just logs those tuples in output queues instead. Fake tuples are identified by adding the second pair of input-tuple pointers towards each tuple. Then the tuple is represented as $T(u)$ also the input stream is termed as S then its identifier or detector became (S, v), it is the most modern tuple which contributes the creation of $T(u)$. These identifiers are also termed as a high watermarks. Though the message is sent between secondary and primary nodes, the watermark is set among downstream and upstream nodes in Fig. 4.

In addition, replication (also named as clustering) is a method to offer high accessibility in distributed and parallel databases. High availability is planned to ensure a continuous service process. Moreover, high availability has two phases. In one phase, it affords a fault-tolerance mechanism by developing redundancy in the way of replication. This means it has multiple replicas or copies of the data from different sites. In some cases the data copies might be crashed, so to maintain the data accessibility and availability is needed to reintroduce the data replicas, thus introducing the new replicas are in a consistent fashion. Subsequently,

Algorithm 2: Fault-tolerant mechanism

```
Function LSR (New tuples)
{
        initialize ()
          current node= resident resource ID
        Current job← getVM job information (currentVM)
        Present fault← get Fault Rate
              LSR (present job, current fault)
              Worktable. New tuple
        If (VM < resourceutilization ) Then
        {
              check nextVM =lighter node()
              lighter node()= less energy hub
              //The job in less energy hub is migrated to the other hub
              Resource=total resource utilization-work
              New tuple=next hub
              // New hub creation
        }end if
Assign security guard as checkpoint
Assign the work with Checkpoint Node
}//end function
```

the transmission process shifts to new replicas and carry on the process. If the heuristics mechanism E_α reports fault occurrence then the replication scheme is processed on the basis of the following steps. Here M_a receives the acknowledgement from the downstream link and new tuples from the link upstream. Hence, the filter functions $S_1[800]$ and creates $F[500]$.

Moreover, M_a spruces its outcome queues at $(T, 50)$ while approaching fresh tuples $T(146)$ and $T(147)$ downstream. This kind of backward plotting is organized by the cause function $((T, u), S) \rightarrow (S, v)$ where (S, v) represents the detector of the tuple $S[u]$. Moreover, the process of replication is to store the data in another site or another node for the recovery process while the transmission node became a fault during the process, which is designed using Algorithm 2. Also to overcome the failure of host the checkpoint strategy is utilized as a rollback mechanism. Moreover, the checkpoint is updated regularly and it has all the copies of processed jobs. Thus, if any host is failed it retrieves the information through its checkpoint.

5 Result and discussion

The proposed research is elaborated in java programming language running in the windows 10 platform. Cloud computing has attracted users with its advancement and great extent; however, due to data security and reliability issues, the users are still afraid of porting tasks to cloud architecture. The users are frightened to post the work in the cloud because of fault occurrence. However, in a cloud computing environment, several hosts and VMs are available to add redundancy in the system, therefore one can migrate the task from the host showing fault to another host. The parameters taken for the proposed work are described in Table 2. Here, multi instructions per second (MIPS).

Table 2 Parameters of proposed work

VM parameters	Specification
Machine name	V medium
RAM (GB)	3.75
Storage	1×4 GB
Bandwidth	1Gbps
Price ($ per hour)	0.0
Physical nodes	200
VM's power (MIPS)	250–1500
Fault probability	0.01
Device Platform	
Operating system	Windows 10
Architecture	X64
Host parameters	6821 MIPS
CPU	2×Intel Xeon E5-2630 2.3 GHz
Memory (DDR3)	128G
Storage capacity	3.81 TB
Montage Sky statistics (astronomy)	
Tasks parameter	Value (fixed)-(min, max, step)
Tasks count ($\times 10^4$)	(2)-(0.4,5,0.4)
Task size ($\times 10^5 MI$)	(1,2) [min, max]
Run time fault: 0.01 s	

5.1 Case study

Let us assume the jobs which were submitted to the cloud grid in priority based order. The assumed jobs have different computational time, execution time, deadline, resource utilization, and different file size. The jobs in VM are carried out by a lot of instructions and rules.

Here the priority-based jobs are identified by the β parameter.

Step 1: Initially α checks the priority work by its work deadline and execution time.

Step 2: The first priority work is represented as β and normal jobs are represented as δ.

Step 3: Then α checks the conditions of all present VM in a cloud environment.

Step 4: Based on the priority, the resources are allocated for each job.

Step 5: Consequently, the deadline and execution time of each job is trained to the AL model.

Step 6: It checks the completed jobs and incomplete jobs based on their particular deadline.

Step 7: If the jobs are not completed by their deadline then it checks free VM and transfers it to that free VM as K_1. The K_1 must have sufficient resources to complete the specified task.

Step 8: If K have more workloads than its capacity, then it is distributed to the other lighter VM.

Step 9: While checking for free VM, if any $K = 0$ then the fault is detected it initiates the tolerant mechanism.

The stable state process in cloud computing is a checkpoint which deals with job and fault consistency. It automatically helps the VM to roll back the client request at any time. Moreover, the checkpoint can update frequently for every few seconds of time, it is more helpful at the time of data recovery. Manual checkpoints are initiated by users to control the date and time. In addition, the parameter alpha monitors the functions of VM regularly, if any one of the VM has more workload than its resource, then it makes the alarm; immediately the fault-tolerant process is initiated. Then the alpha is predicted by the VM and distributed the loads to another lighter VM, which is detailed in Fig. 5. In addition, the fault-tolerant mechanism helps to form the new tuple to maintain the failed host.

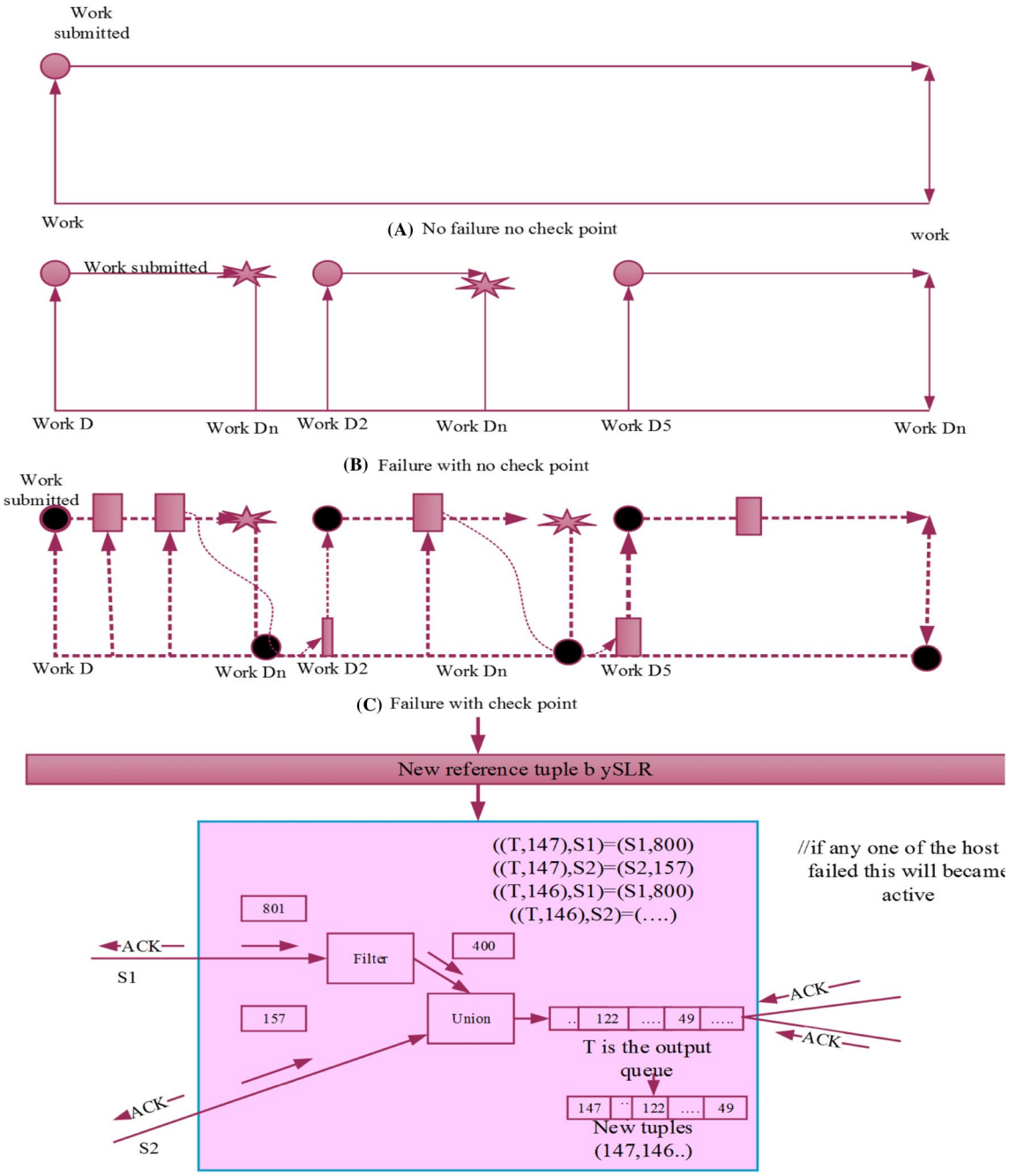

Fig. 5 Check point with fault-tolerant SLR

The results of the case study are obtained in Fig. 6a describes job versus throughput in the millisecond it is based upon job transferring rate. Before assigning the job to one VM its workload and execution time must be evaluated, it is more helpful to assign the priority-based work to the specified VM.

After the job migration process, the attacks such as fake data injection, DOS and reply attack was launched in the job transmission VM channel to estimate the reliability of the system. But, the VM channel remains secure because of the ant lion fitness function in the designed proposed model. Here, the VM and user information's are already monitored by the ant lion, so if any attacks have been interrupted in VM's medium then it immediately blocks that VM or users by their hunting fitness.

Thus the proposed HGW-ALM with LSR has gained the finest privacy range, which is detailed in Fig. 6.

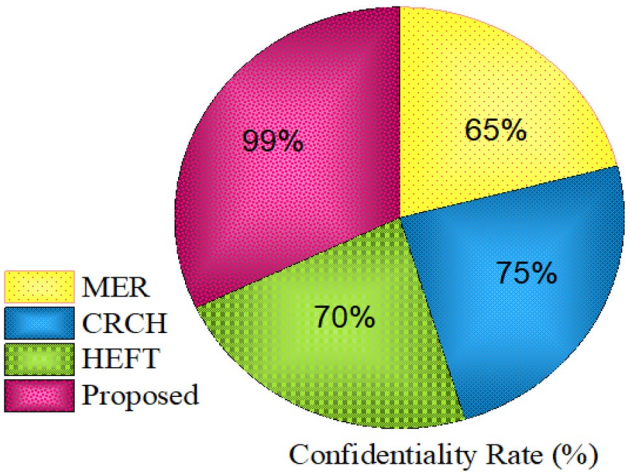

Fig. 6 Validation of confidential rate

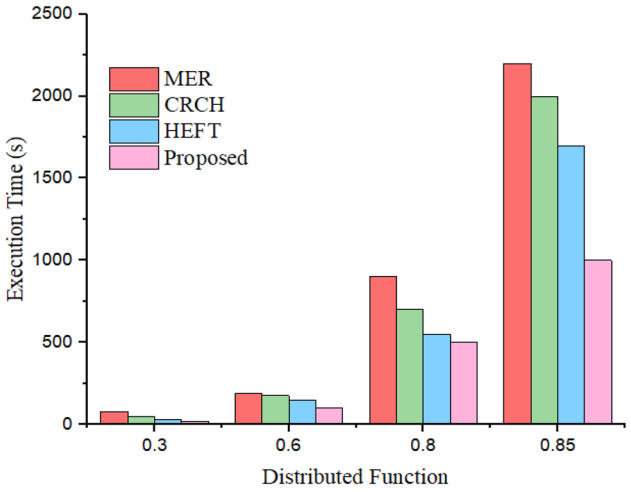

Fig. 7 Execution time (s) assessment

5.2 Performance metrics

The present research developed an efficient scheduling and fault-tolerant mechanism by hybrid algorithm models, the graphical representation proved the efficiency of the proposed method by achieving a better result. In addition, to evaluate the efficiency of the proposed strategy some of the recent existing approaches are adopted such as maximum effective reduction (MER) (Lee et al. 2015) and checkpointing replication based on clustering heuristics (CRCH) (Lee and Gil 2019), heterogeneous earliest finish time (HEFT) (Setlur et al. 2019).

HEFT is a heuristic model and provides a better result for the scheduling process. This process is utilized to reduce the job execution time and maximize resource utilization. MER minimized the execution time and resource usage by the consistent workflow schedule. The job is scheduled by the nearer VM selection, thus it reduces the complexity in job execution. Moreover, it is applicable in all environments and deals with all workload conditions.

The CRCH mechanism has three modules such as checkpointing, replication, and clustering. The clustering phase calculates the computation time of each job. The replication module is utilized to replicate CRCH the data in cloud computing. To compare the performance of the current research model some of the important parameters should be evaluated such as execution time, response time, delay time, makespan time, and distributed function.

5.2.1 Execution time

The execution time for each job is different because each job has different workloads. Thus the execution time is defined as taken to complete the assigned task. Moreover, the performance of the VM is evaluated using its execution time.

Table 3 Distributed function versus execution time

Distributed function	Distributed function versus execution time[a]			
	MER	CRCH	HEFT	Proposed (HGW–ALM with LSR)
0.3	80	50	30	20
0.6	190	175	150	100
0.8	900	700	550	500
0.85	2200	2000	1700	1000

[a]Seconds

The assumption of job execution time is initially calculated by the grey wolf and ant lion mechanism based on that the jobs are assigned. Before the job allocation, the presence of each VM or node is calculated with the chief parameter as node weight and distance.

The execution time of an assigned job is detailed in Fig. 7 and Table 3. Moreover, the job execution time is based on the job scheduling mechanism. Here heuristic algorithm is used to schedule the jobs based on priority wise, thus the accurate and proficient job scheduling process results in good execution (less) time. Hence the figure illustrated the distributed jobs with execution time. The time taken by the proposed model is 1000s for 0.85 distributed functions. Here, job completion time is considered as execution time, here the execution time gets differed in Fig. 7, this is because of delay limit and fault.

5.2.2 Response time

After the job allocation, the first reply is considered as response time. It is based on the time taken by VM to initiate

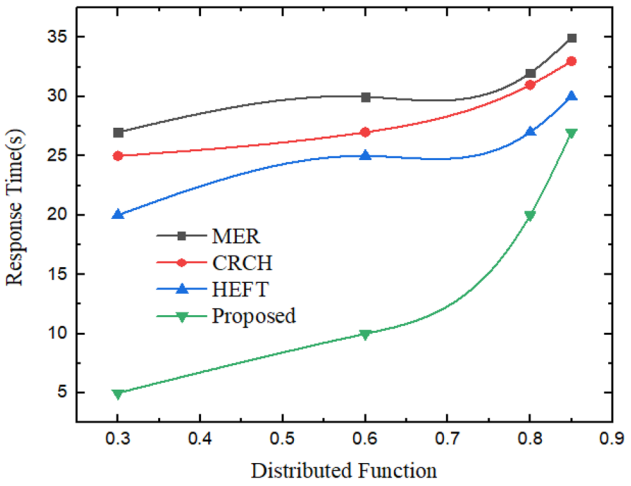

Fig. 8 Response time validation

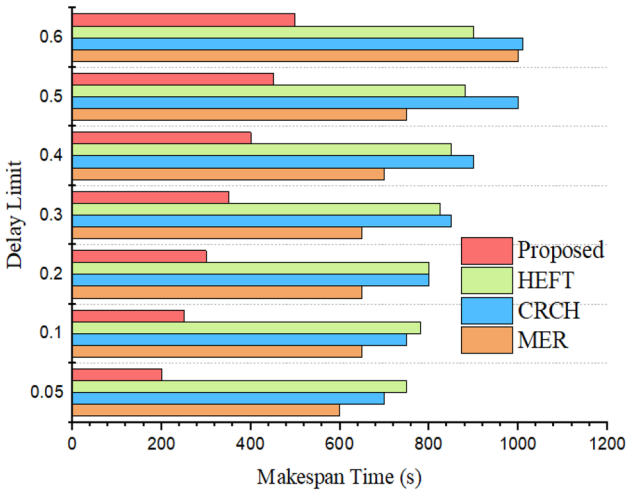

Fig. 9 Assessment of delay time vs. makespan time

Table 4 Response time comparison

Response time (s[a])				
Distributed function	MER	CRCH	HEFT	Proposed (HGW–ALM with LSR)
0.3	27	25	20	5
0.6	30	27	25	10
0.8	32	31	27	20
0.85	35	33	30	27

[a]Seconds

the process. The time taken to initiate the process by the current proposed model is 5s. In a cloud computing framework, the response time is calculated based on the amount of interval from when a request was submitted till the first reply was produced that is elaborated in Fig. 8 and Table 4.

One of the chief aspects in the cloud task scheduling model is a load balancing strategy. The task load is equally shared to other VM then it increases the resource efficiency and reduces the response time. In this present research work, the task load is evaluated by the finest function of the GW model then the task is shared by the nearer neighboring node by Ant Lion fitness. Here, the response time was varied based on the different workloads; if the job count was less then short time is sufficient for the response.

The fitness function of the HGW–AL model is utilized to investigate the status of servers and to allocate the job in a priority manner, also if any issues are present in the VM then it alert that a specific VM is in trouble. Usually, the task allocation strategy is created to reduce the job execution time and to save the balance energy of nodes to maintain the network lifetime and load to ensure the job wouldn't be cracked by a sudden failure of hubs.

5.2.3 Makespan time

Makespan is demarcated as the time taken to complete the specified task from the beginning to end for all assigned jobs. In addition, for the multi-target appliance, there are several targets are assigned to monitor the specified area. Once the target is assigned, reliable tracking is needed to track the job handling hubs because if the node energy is drained new tuples want to be created to continue the task without any interruption. So the calculation of job weight is an important parameter for assigning the job. Depends upon the delay limit, the makespan time can be varied. Thus Fig. 9 has shown the variation in makespan time.

The determination of work starting and ending time is termed as makespan time also the deadline crossed work is validated under delay limit in Fig. 9 and Table 5. Thus, the makespan time with delay time comparison is elaborated

Table 5 Delay time vs. makespan time

Delay limit	Delay time versus makespan time (s)			
	MER	CRCH	HEFT	Proposed (HGW–ALM with LSR)
0.05	600	700	750	200[a]
0.1	650	750	780	250
0.2	650	800	800	300
0.3	650	850	825	350
0.4	700	900	850	400
0.5	750	1000	880	450
0.6	1000	1010	900	500

[a]Makespan time of proposed model

Table 6 Delay limit versus resource usage

Delay limit	Delay limit versus resource usage[a]			
	MER	CRCH	HEFT	Proposed (HGW–ALM with LSR)
0.05	6	7	7.5	2.1
0.1	6.5	7.5	7.8	2.5
0.2	6.5	8	8	3
0.3	6.5	8.5	8.25	3.5
0.4	7	9	8.5	4
0.5	7.5	10	8.8	4.5
0.6	10	10.1	9	5

[a]Bit per second

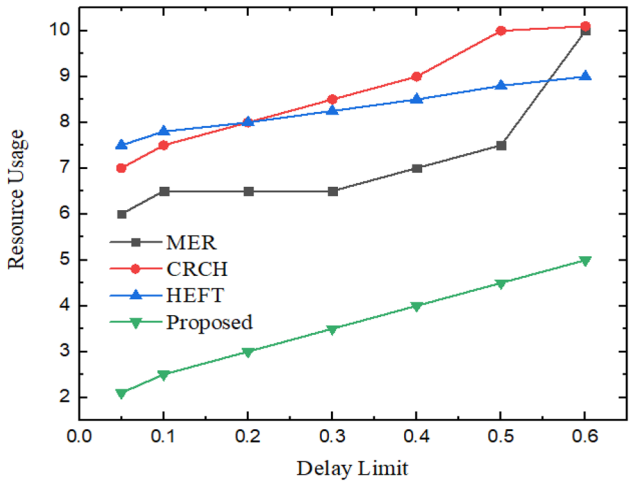

Fig. 10 Average resource usage

Table 7 Scheduling time

Methods[a]	Scheduling time (s)[b]
MER	800
CRCH	600
HEFT	400
Proposed (HGW–ALM with LSR)	200

[a]Existing techniques

[b]Task allocating time

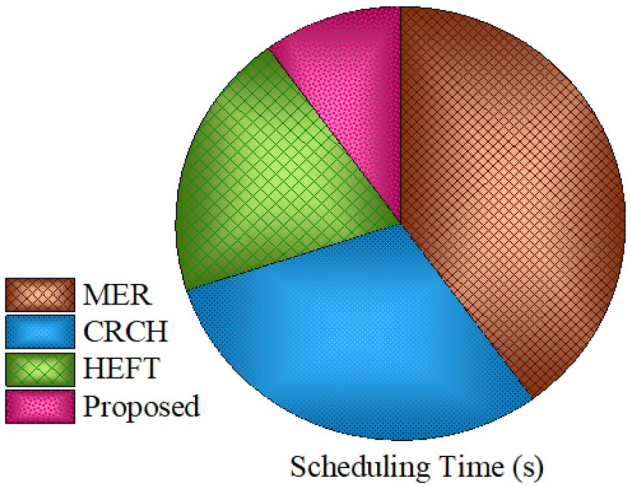

Fig. 11 Comparison of scheduling time

in Fig. 9, here the proposed strategy attained a delay limit as 420s.

5.2.4 Resource usage

In the cloud network, tasks are tangible-time independent jobs without any model constraints, because the jobs which have model constraints are correspondent to independent jobs by arriving periods and deadlines of the specified tasks. All the hubs have backup copies to retrieve the information when any one of the nodes is failed.

The process needs some resources to complete the task which is validated as resource usage; it is graphically shown in Fig. 10 and Table 6. The proposed scheme consumed less CPU utilization compared to existing approaches that means HGWALM has consumed CPU utilization ratio as 0.5 % (max). Based upon the delay limit of each work completion, the resource utilization gets varied. If the method has taken more time to complete the process then it has required more resources.

5.2.5 Scheduling time

The task scheduling time is defined as the total execution time divided by specified work execution time t it is defined by Eq. (7) the scheduling time is calculated on the basis of work assigning time to each VM or hub.

$$\text{Scheduling time} = \frac{\textit{Total execution time}}{t} \tag{7}$$

The cloud process of end users can reduce the available sources, as per the demands for each specified application. Moreover, the cloud is the service provider that means any time any user can capable to request the services. Thus the scheduling process is described in Fig. 11 and Table 7.

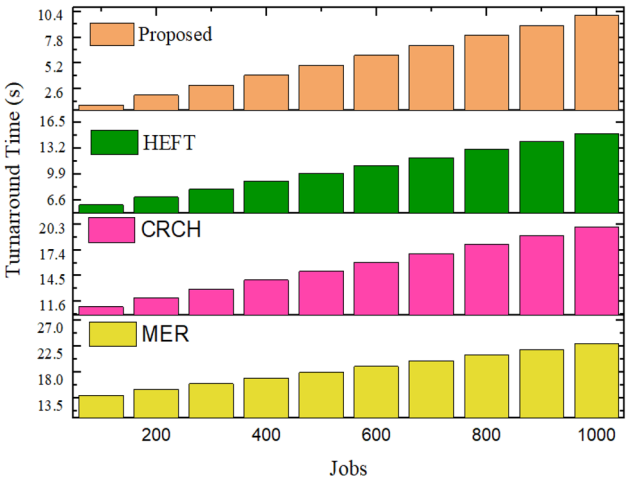

Fig. 12 Execution time comparison

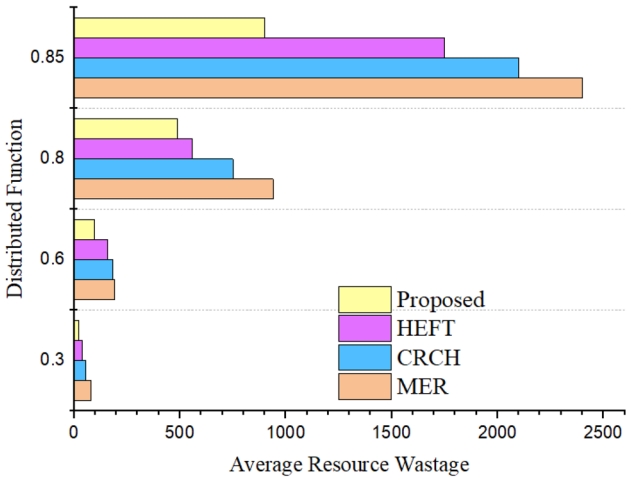

Fig. 13 Average resource wastage

Table 8 Comparison of turnaround time (s)

Jobs	MER	CRCH	HEFT	Proposed (HGW–ALM with LSR)
100	14	11	6	1[a]
200	15	12	7	2
300	16	13	8	3
400	17	14	9	4
500	18	15	10	5
600	19	16	11	6
700	20	17	12	7
800	21	18	13	8
900	22	19	14	9
1000	23	20	15	10

[a]Processing time of 100 jobs

Based upon the algorithm robustness, the scheduling time gets varied.

5.2.6 Turnaround time (ms)

The failure that occurs during transmission is defined as turnaround time, which is evaluated by the difference of job execution time and arrival time in Eq. (8),

$$\textit{Turnaround time} = \textit{execution time} - \textit{arrival time} \tag{8}$$

During the process, the host failure occurs then LSR maintains the host by creating the new tuple. Thus, the proposed method achieved less failure time as 10s for 1000 jobs. Finally, the scheduled job execution time achieved by the proposed work and their comparisons are detailed in Fig. 12 and Table 8. The turnaround time is dependent on the algorithm rapidity for the scheduling and fault-tolerant process. Thus, the proposed model has attained less execution time than other models. Also, when the job size was increased then execution time also get vary from the initial stage.

Table 9 Average resource wastage

Distributed function	Average resource wastage[a]			
	MER	CRCH	HEFT	Proposed (HGW–ALM with LSR)
0.3	82	55	40	21
0.6	193	185	160	98
0.8	940	750	560	490
0.85	2400	2100	1750	900

[a]Bit per second

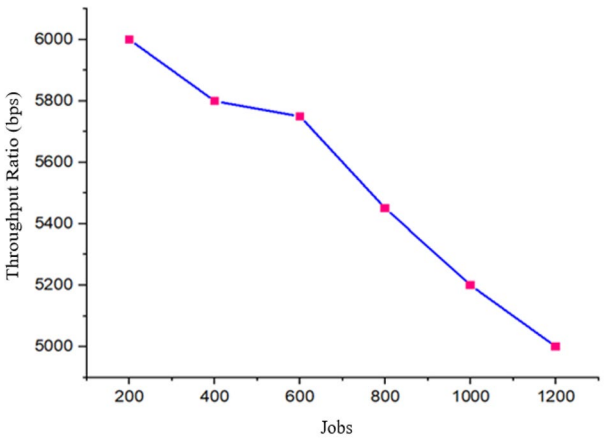

Fig. 14 Throughput ratio (bps)

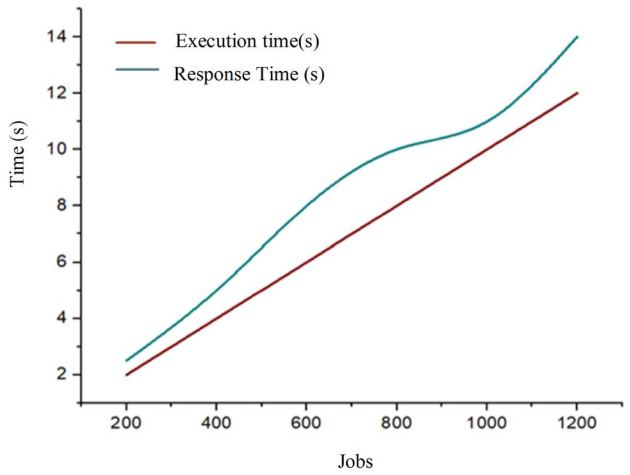

Fig. 15 Execution and response time (s)

5.2.7 Average resource wastage

The additional usage of resources, which are taken to complete the process is termed as resource wastage. The resource wastage is calculated on the basis of extra resources needed to complete the specified task. Based on the algorithm complexity, the resource wastage has differed. By the validation, the proposed model has reported very less resource wastage because of algorithm flexibility. If the workloads are increased then the resource wastage measure also increased from the previous one.

The graphical representation shows that MER attained the maximum resource wastage as 2400, CRCH pertained 2100 resource wastage, HEFT attained 1750 and the proposed model achieved less resource wastage as 900, which is described in Fig. 13 and Table 9.

5.3 Discussion

So the proposed model enhanced the performance of job scheduling and fault-tolerant scheme in a cloud environment. The obtained throughput ratio for our developed model is shown in Fig. 14. Moreover, the high throughput measure attained a better data transmission rate.

The response time in the cloud paradigm is validated by the time taken to accept the request after the job allocation. Thus the response and execution time of the proposed approach is shown in Fig. 15. On the other hand for the performance evaluation, Hybrid Genetic and Particle Swarm Optimization and Artificial Bee Colony model also taken for an evaluation, which is detailed in Table 10.

Thus the outcome of the projected methodology proved that the cloud is enhanced and capable to process all kinds of works.

5.3.1 Computation complexity

In most cases, the effective score of the proposed model was depended on its time complexity. The complexity of time is calculated by noting the run time of each algorithm with its own functions.

Table 10 Performance comparison

Technique	Method	Merits	Demerits
MER	In used maximum resource to obtain the less makespan time	It found the optimal point to assign the job	The wastage of the resource is very high
CRCH	It actively handles the jobs by the checkpointing strategy	It decreased the resource usage	Time complexity
HEFT	the light mass matched checkpointing facilitates an efficient reassign of failed jobs also it ensures the workflow conclusion even in unstable environments	It minimized the wastage of resource	In an unstable environment, securing the VM is very difficult
HGPSO (Kumar and Venkatesan	All the jobs are arranged on the basis of precedence and appropriate resources are owed to finish the job	In the meantime, the New jobs are estimated and stored in an on-demand queue	In the presence of faults, its performance is insufficient
Enhanced ABC (Thanka et al. 2019)	It arranges the scheduling order by its fitness function	It improved the quality of service in cloud computing	It takes more time for execution
Proposed (HGW–ALM with LSR)	Jobs are scheduled with the help of the GW fitness model and the fault is predicted by the AL fitness. Finally, the faults are tolerated using the LSR approach	Jobs are completed with high security and in less execution time	It takes few seconds to Design the HGW–ALM and LSR in the cloud paradigm

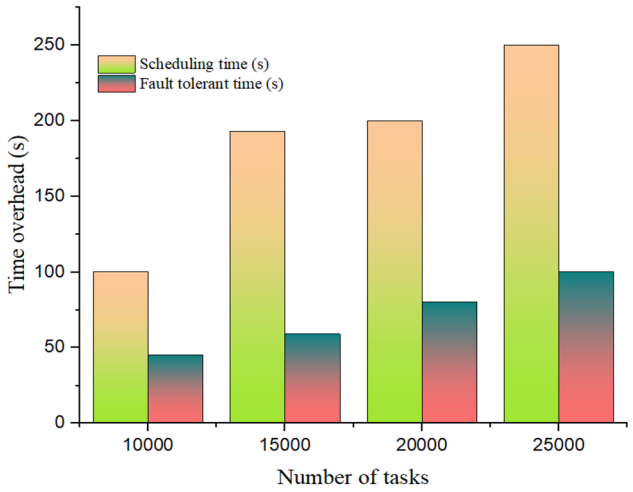

Fig. 16 Time complexity

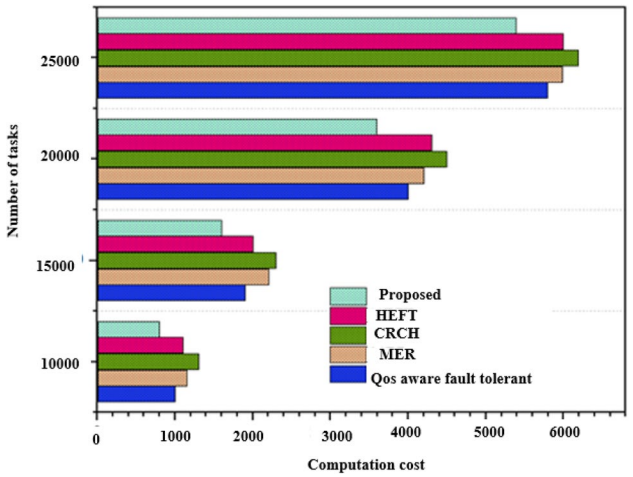

Fig. 17 Comparison of computation cost

Here, the scheduling time is validated using a hybrid optimization model as HGW–ALM, and the time of fault-tolerant was evaluated using the LSR strategy. Moreover, the methods which have offered the best solution in a short duration are termed as appropriate models. Hence the time utilized for scheduling and the fault-tolerant process is shown in Fig. 16.

The computation model is calculated to verify the successive score of the proposed model; here, the proposed model has diminished the cloud services space. Hence, the comparison of computation cost is detailed in Fig. 17.

Furthermore, comparison validation with several parameters proved the efficiency of the proposed work in an efficient way. Moreover, job migration and checkpoint strategy have reduced the average resource wastage in a tremendous way. Thus the proposed strategy is suitable for job allocation and fault-tolerant strategy.

6 Conclusion

Job scheduling and fault-tolerant in the cloud environment is more crucial, this current research focuses on a novel Hybrid heuristics algorithm as HGWALM for scheduling purposes and a novel hybrid fault-tolerant mechanism LSR is developed to tolerate the fault during the job running or processing time. Moreover, the fault-tolerant strategy is more important in the scheduling process because one server can handle a huge number of jobs at a single time. During the process, if any one of the program modules fails it tends to delay and spoil all the jobs which are currently in the process. So the effective tolerant mechanism as LSR scheme is used to tolerate the faults, simultaneously if any fault occurs then the current process is migrated to another server by HGWALM. In addition, the robustness of the proposed algorithm is checked against different attacks like fake data injection, replay attack and DoS. Moreover, the comparison results proved the efficiency of the proposed work, by showing the improvement of 95% scheduling time reduction. Hence, the consumed maximum job execution time is 400 s for 1.8 distributed functions.

Declarations

Conflict of interest The authors declare that they have no potential conflict of interest.

Human and animal rights All applicable institutional and/or national guidelines for the care and use of animals were followed.

Informed consent For this type of study formal consent is not required.

References

Abedinia O, Zareinejad M, Doranehgard MH et al (2019) Optimal offering and bidding strategies of renewable energy based large consumer using a novel hybrid robust-stochastic approach. J Clean Prod 215:878–889. https://doi.org/10.1016/j.jclepro.2019.01.085

Ahmad Z, Jehangiri AI, Iftikhar M, Umer AI (2019) Data-oriented scheduling with dynamic-clustering fault-tolerant technique for scientific workflows in clouds. Program Comput Softw 45(8):506–516. https://doi.org/10.1134/S0361768819080097

Ahmad Z, Nazir B, Umer A (2021) A fault-tolerant workflow management system with quality-of-service-aware scheduling for scientific workflows in cloud computing. Int J Commun Syst 34(1):e4649. https://doi.org/10.1002/dac.4649

Bagal HA, Soltanabad YN, Dadjuo M et al (2018) Risk-assessment of photovoltaic-wind-battery-grid based large industrial consumer using information gap decision theory. Sol Energy 169:343–352. https://doi.org/10.1016/j.solener.2018.05.003

Bhushan K, Gupta BB (2019) Distributed denial of service (DDoS) attack mitigation in software defined network (SDN)-based cloud computing environment. J Ambient Intell Humaniz Comput 10(5):1985–1997. https://doi.org/10.1007/s12652-018-0800-9

Bicakci K, Tavli B (2009) Denial-of-Service attacks and countermeasures in IEEE 802.11 wireless networks. Comput Stand Interfaces 31(5):931–941. https://doi.org/10.1016/j.csi.2008.09.038

Ding Y, Yao G, Hao K (2017) Fault-tolerant elastic scheduling algorithm for workflow in cloud systems. Inf Sci 393:47–65. https://doi.org/10.1016/j.ins.2017.01.035

El Makkaoui K, Beni-Hssane A, Ezzati A (2019) Speedy cloud-RSA homomorphic scheme for preserving data confidentiality in cloud computing. J Ambient Intell Humaniz Comput 10(12):4629–4640. https://doi.org/10.1007/s12652-018-0844-x

Gao W, Darvishan A, Toghani M, Mohammadi M et al (2019) Different states of multi-block based forecast engine for price and load prediction. Int J Electr Power Energy Syst 104:423–435. https://doi.org/10.1016/j.ijepes.2018.07.014

Gao J, Wang H, Shen H (2020) Task failure prediction in cloud data centers using deep learning. IEEE Trans Serv Comput. https://doi.org/10.1109/TSC.2020.2993728

Ghadimi N, Akbarimajd A, Shayeghi H, Abedinia O (2018) Two stage forecast engine with feature selection technique and improved meta-heuristic algorithm for electricity load forecasting. Energy 161:130–142. https://doi.org/10.1016/j.energy.2018.07.088

Hosseinzadeh M, Sinopoli B, Garone E (2019) Feasibility and detection of replay attack in networked constrained cyber-physical systems. 2019 57th annual allerton conference on communication, control, and computing (Allerton). IEEE. https://doi.org/10.1109/ALLERTON.2019.8919762

Khaldi M, Rebbah M, Meftah B, Smail O (2020) Fault tolerance for a scientific workflow system in a cloud computing environment. Int J Comput Appl 42(7):705–714. https://doi.org/10.1080/1206212X.2019.1647651

Khodaei H, Hajiali M, Darvishan A, Sepehr M et al (2018) Fuzzy-based heat and power hub models for cost-emission operation of an industrial consumer using compromise programming. Appl Therm Eng 137:395–405. https://doi.org/10.1016/j.applthermaleng.2018.04.008

Kumar AMS, Venkatesan M (2019) Task scheduling in a cloud computing environment using HGPSO algorithm. Clust Comput 22(1):2179–2185. https://doi.org/10.1007/s10586-018-2515-2

Latiff MSA, Madni SHH, Abdullahi M (2018) Fault tolerance aware scheduling technique for cloud computing environment using dynamic clustering algorithm. Neural Comput Appl 29(1):279–293. https://doi.org/10.1007/s00521-016-2448-8

Lee JH, Gil JM (2019) Adaptive fault-tolerant scheduling strategies for mobile cloud computing. J Supercomput 75(8):4472–4488. https://doi.org/10.1007/s11227-019-02745-5

Lee YC, Han H, Zomaya AY, Yousif M (2015) Resource-efficient workflow scheduling in clouds. Knowl Based Syst 80:153–162. https://doi.org/10.1016/j.knosys.2015.02.012

Li C, Tang J, Ma T, Yang X, Luo Y (2020) Load balance based workflow job scheduling algorithm in distributed cloud. J Netw Comput Appl 152:102518. https://doi.org/10.1016/j.jnca.2019.102518

Liu Y, Ning P, Reiter MK (2011) False data injection attacks against state estimation in electric power grids. ACM Trans Inf Syst Secur (TISSEC) 14(1):1–33. https://doi.org/10.1145/1952982.1952995

Mirjalili S (2015) The ant lion optimizer. Adv Eng Softw 83:80–98. https://doi.org/10.1016/j.advengsoft.2015.01.010

Mirjalili S, Mirjalili SM, Lewis A (2014) Grey Wolf optimizer. Adv Eng Softw 69:46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007

Mukherjee P, Swain T, Datta A (2020) Issues of some task scheduling strategies on sensor cloud environment. Smart intelligent computing and applications. Springer, Singapore, pp 651–663

Park DS (2018) Future computing with IoT and cloud computing. J Supercomput 74(12):6401–6407. https://doi.org/10.1007/s11227-018-2652-7

Ponmagal RS, Karthick S, Dhiyanesh B, Balakrishnan S, Venkatachalam K (2020) Optimized virtual network function provisioning technique for mobile edge cloud computing. J Ambient Intell Humaniz Comput. https://doi.org/10.1007/s12652-020-02122-8

Qiang W (2019) Performance and security in cloud computing. J Supercomput 75(1):1–3. https://doi.org/10.1007/s11227-018-2671-4

Rezaeipanah A, Mojarad M, Fakhari A (2020) Providing a new approach to increase fault tolerance in cloud computing using fuzzy logic. Int J Comput Appl. https://doi.org/10.1080/1206212X.2019.1709288

Rodriguez MA, Buyya R (2014) Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds. IEEE Trans Cloud Comput 2(2):222–235. https://doi.org/10.1109/TCC.2014.2314655

Saeedi M, Moradi M, Hosseini M et al (2019) Robust optimization based optimal chiller loading under cooling demand uncertainty. Appl Therm Eng 148:1081–1091. https://doi.org/10.1016/j.applthermaleng.2018.11.122

Sarmila GP, Gnanmbigai N, Dinadayalan P (2019) Self scheduling based on hexagonal Chebysev Gaussian and discrete time organized mapping in cloud. 2019 International conference on smart systems and inventive technology (ICSSIT). IEEE. https://doi.org/10.1109/ICSSIT46314.2019.8987946

Setlur AR, Nirmala SJ, Singh HS, Khoriya S (2019) An efficient fault tolerant workflow scheduling approach using replication heuristics and checkpointing in the cloud. J Parallel Distrib Comput. https://doi.org/10.1016/j.jpdc.2019.09.004

Suliman YM, Yousif A, Bashir MB (2019) Shark smell optimization (SSO) algorithm for cloud jobs scheduling. International Conference on Computing. Springer, Cham. https://doi.org/10.1007/978-3-030-36368-0_7

Tamilvizhi T, Parvathavarthini B (2019) A novel method for adaptive fault tolerance during load balancing in cloud computing. Clust Comput 22(5):10425–10438. https://doi.org/10.1007/s10586-017-1038-6

Thanka MR, Maheswari PU, Edwin EB (2019) An improved efficient: artificial bee colony algorithm for security and QoS aware scheduling in cloud computing environment. Clust Comput 22(5):10905–10913. https://doi.org/10.1007/s10586-017-1223-7

Wu B, Hao K, Cai X, Wang T (2019a) An integrated algorithm for multi-agent fault-tolerant scheduling based on MOEA. Future Gener Comput Syst 94:51–61. https://doi.org/10.1016/j.future.2018.11.001

Wu N, Zuo D, Zhang Z (2019b) Dynamic fault-tolerant workflow scheduling with hybrid spatial-temporal re-execution in clouds. Information 10(5):169. https://doi.org/10.3390/info10050169

Yan H, Zhu X, Chen H, Guo H, Zhou W, Bao W (2019) DEFT: dynamic fault-tolerant elastic scheduling for tasks with uncertain runtime in cloud. Inf Sci 477:30–46. https://doi.org/10.1016/j.ins.2018.10.020

Yao G, Ren Q, Li X, Zhao S (2020) A hybrid fault-tolerant scheduling for deadline-constrained tasks in Cloud systems. IEEE Trans Serv Comput. https://doi.org/10.1109/TSC.2020.2992928

Zhu X, Wang J, Guo H, Zhu D, Yang LT, Liu L (2016) Fault-tolerant scheduling for real-time scientific workflows with elastic resource provisioning in virtualized clouds. IEEE Trans Parallel Distrib Syst 27(12):3501–3517. https://doi.org/10.1109/TPDS.2016.2543731