Enhanced flower pollination algorithm based security and QoS aware scientific workflow scheduling on inter cloud

[1]Arravinth J, [2]Dr. D. Manjula, [3]Dr. AR. Arunarani

[1]Research Scholar, [2]Professor, [3]Teaching Follow
Department of Computer Science and Engineering
College of Engineering Guindy, Anna University, Chennai

***Abstract*: Because of its multifaceted benefits in adapting cloud computing to real-world science workflow applications, we intend to use cloud computing to implement scientific workflows. In the present work, we aim to plan workflows on measurable resources in the cloud. Scheduling workflows to the appropriate source here is an NP hard problem. Ensuring its security is an important criterion as scheduling takes place at the border of third parties. To provide QoS such as makespan, cost, during the schedule and enforcing the security are the significant objective of the proposed work. To achieve this objective enhanced flower pollination algorithm is proposed. The main use of this is to solving global optimization problems very fastly as well as, this algorithm perfectly suitable for parallel processing and well capable of making trade-off among intensification and diversification. The performance of proposed algorithm analysed in terms of makespan, and security.**

***Keywords*: Task scheduling, workflow, makespan, enhanced flower pollination algorithm, security.**

1. Introduction

Cloud computing [1] is an aggregated system node which delivers application, CPU, bandwidth, security, and various computing capabilities. It provides pay-per-use on-demand activities through the network [2]. To develop as well as manage their cloud processing infrastructures, virtualizing solutions use cloud computing [3]. It allows several clients to share a single application or physical resource, and virtualization can handle load balancing [4]. The information technology, cloud computing paradigm depends primarily on the ease and speed with which IT resources can be assigned [5], relieving end-users from IT infrastructure and location issues. All of this is available on a pay-per-use basis [6]. Cloud environments enable service providers and internet providers. The service provider is responsible for the application, system, and infrastructure used to execute the activity [7]. Internet service providers are considered to be Cloud clients or customers at the identical period. The most significant strategy for using cloud security is work scheduling [8]. To decrease makespan without violating precedence requirements, the scheduling algorithm should distribute tasks depending on available cloud resources.

In the cloud, effective task scheduling would make use of all accessible resources to boost the efficiency of the transfer system [9]. This raises the difficulty of scheduling decisions in order to ensure task transfer efficiency [10]. As a result, the task transfer scheduling decision is compounded by the challenge of addressing transfer efficiency [11]. It assigns user tasks to cloud resources to maximize utilization, reduce make span, and balance cloud infrastructure to avoid overburdening activities [12, 13]. Static or dynamic scheduling options are available in task scheduling. The scheduler defines the specifics of the resources and tasks in static planning. The details of tasks and resources are undetermined from the start in dynamic scheduling [14, 15]. The scheduler creates dynamic scheduling plans to determine which resources are suited for user activity [16]. Resource providers, task/service scheduling, and clients are all important aspects of cloud technology. The duration for which a resource is assigned to a request is known as scheduling [17]. Scheduling algorithms are in charge of assigning resources to task requests in the Cloud. The task scheduling is the most critical aspect of cloud security for securing data in the cloud [18].

2. Literature Review

Zhu, Q.H., et al, [19] have analyzed task scheduling method called matching and multi-round allocation (MMA) to reduce the time and cost of any tasks that are subject to protection as well as durability restrictions. CloudSim made use of the modified cuckoo search (MCS), hybrid chaotic particle search (HCPS), modified artificial bee colony (MABC), max-min, and min-min algorithms. Shorter makespan, cost reduction, more resource utilization, and a good trade among duration and financial impact are all achieved as a result of the implementation.
Pradeep, K. and Jacob, T.P., [20] have developed a multi objective task scheduling in cloud security using the hybridization of Cuckoo Search and Gravitational Search algorithm. In comparison to a single objective function, a multi-objective optimization strategy was utilized to increase scheduling efficiency. The algorithm balances cost, energy, and resource usage based on the end-needs. user's The experiment's outcome is a low-cost, low-energy solution.
Parmeet Kaur and Shikha Mehta, [21] have displayed an asset provisioning and workflow scheduling for cloud condition utilizing an expanded Shuffled Frog Leaping Algorithm. A revision to the meta-heuristic calculations was analyzed with the purpose of making the resultant organization toll optimal while still meeting the submission condition. The simulation results show a significant reduction in the performance requirements of attaining the lowest possible processing cost and fulfilling deadlines. When compared

to the other algorithms studied, such as PSO and SFLA, this technique was able to lower the overall execution cost by up to 77 percent.

Fellir, F., et al, [22] have developed task scheduling in a cloud-fog computing platform, using a multi-agent based model. Its goal was to serve the most critical task first, taking into account the task's priority, wait time, status, and resources needed to perform it properly. Additionally, the author analyzed a modification to the task's priority value throughout the scheduling phase, while considering into account the task's dependencies on other tasks and their priorities. The results of simulations suggest that approach can improve resource consumption and performance.

Sharma and Jain [23] have analyzed task scheduling in cloud security using enhanced Ant Colony Optimization (EACO) algorithm. This algorithm primarily aids in the reduction of overall finishing duration for task scheduling on resources. It was accomplished by dividing the ordered delivered tasks into groupings - the task sub list. By using CloudSim toolbox, the EACO algorithm was generated as well as compared to an existing nature-inspired algorithm. The findings of the experiment suggest that reducing the time and cost of production was possible.

Tawfeek, M.A. and Elhady, G.F., [24] have developed dynamic tasks scheduling over cloud's in security using hybrid algorithm. It combines the characteristics of three efficient swarm computing strategies for finding a near-optimal solution to complex multidimensional challenges. It takes advantage of the advantages of ant colony behavior, particle swarm behavior, and honeybee foraging behavior. Simulation findings show that the hybrid algorithm was superior, achieving high resource utilization and outperforming the basis of makespan and degree of imbalance by a wide margin.

Abdullahi, M. and Ngadi, M.A., [25] have analyzed scheduling of tasks on cloud resources using discrete Symbiotic Organism Search (DSOS) algorithm. SOS simulates the interdependent connections that species in an environment show (mutualism, commensalism, and parasitism). When the investigation becomes larger, DSOS converges faster, making it suited for large-scale scheduling challenges. The effectiveness of DSOS was significantly better than that of PSO, according to a t-test analysis of the method was especially true for large search spaces.

3. Problem formation

Various categories of virtual machines are available from the cloud operator. Table 1 shows the various parameters for every VM. The physical machines (PMs) or hosts are made up of various numbers of virtual machines (VMs), which are divided into three categories: green, yellow, and red zones. Every zone must maintain its specific degrees of protection. On the cloud provider section, security guaranteed level (SGL) is utilized to show the proportion of protection that may be offered to VMs. Various SGL limitations are supported by three various sorts of zones. We adjust the SGL ranges [0.9, 1], [0.70, 0.89] and [0.4, 0.69] for green zone, yellow zone and red zones, respectively. SGL refers to the three characteristics of security namely, authentication, confidentiality, and integrity. Therefore, the SGL equation of a VM_j other than the other specification parameters is described as given in (1).

$$SGL_{VM_j} = Average\{SGL^A, SGL^C, SGL^I\} \quad (1)$$

The SGL values are lies between [0-1]. Different SGL values can be achieved by implementing the corresponding algorithms given in Table 2 for confidentiality [1].

Cryptographic algorithms	SGL	Computation time (ms)
SEAL	0.08	168.75
RC4	0.14	96.43
Blowfish	0.36	37.5
Knufu/Khafre	0.40	33.75
RC5	0.46	29.35
Rijndael	0.64	21.09
DES	0.90	15
IDEA	1.00	13.5

Table 1: Cryptographic algorithms for confidentiality and its SGL [26]

On the other hand, we use another parameter used in the user perspective, namely the security demand (SD). As with SGL, the security requirement is usually denoted by a value of [0,1] in the interval. The SD is obtained by the provider as part of the SLA that is negotiated between the provider and the user. When the operator isn't concerned regarding the protection of several variables, they can express it as a single-valued variable. In a certain instance, all three variables would be given the identical weight amount.

$$SD_{ti} = Avg\{WT_A, WT_C, WT_I\} \quad (2)$$

Equation (2) represents the weight of the SD for each security parameter authentication, confidentiality and integrity as WT_A, WT_C and WT_I respectively.

4. Objective function formulation

The goal of the work is use a metaheuristic approach to schedule tasks on the cloud in the most efficient way possible. Consider the workflow model as a designed acyclic graph, $W = (T, E)$. Here, T represent n number of tasks it is $t_i \in T$ defined task for the processing component. E represent the edges among tasks and it $e(i, j) \in E$ establishes the process dependency condition, stating that activity T_i must finish its function until process T_j starts.

The $Parent(t_i)$ is denoted as predecessors and $child(t_i)$ denoted as successors of task t_i. In every workflow paradigm, there is an entering activity with an exiting activity. The starting task of the application is entry task T_{entry} which has no predecessors, while the final task is called as exit task T_{exit} with no successors.

The purpose of our recommended approach is to reduce total processing cost (TPC) and total processing time (TPT) to satisfy the user's security needs. This approach focused to schedule a task based on multi-objective model. The multi-objective function balances the makespan and cost based on weight factor ω. The fitness of proposed approach is given in equation (3)

$$\begin{aligned} F &= Min\{\omega * TPC + (1-\omega) * TPT\} \\ &Subjected to\, SD_{t_i} \le SGL_{VM\,j} \end{aligned} \quad (3)$$

TPC and TPT are the set of values associated with the sorting of different VMs for the required SD of the i^{th} task used with the required SGL in the j^{th} VM. The computation duration connected with the suitable SGL is taken into account while determining the goal component.

5. Security risk analysis

We evaluate the security of our proposed system because the cloud environment is vulnerable. We define probability of security in scheduling a task t_i to a vm_j as $P_{Security}(t_i, vm_j)$.

The security of a work process in the VM is increased by reducing the differential among the task's protection criteria and the VM's protection quality. Such shift is in line with the high-speed transmission. This is described in equation (4) as a component of the distinction between SD and SGL, which determines the possibility of protection. The protection hazard parameter is expressed as a fractional quantity. The critical coefficient implies that as the divergence among SD_{ti} and SGL_{vmj} increases, so does the rate of failure. A process rejection at a facility may be caused by VM hijacking, VM robbery, Super Jacking, a significant system assault, or a security-imposed barricade that prevents connection.

$$P_{Security}(t_i, vm_j) = \begin{cases} 0, & SD_{ti} \ge SGL_{vmj} \\ 1 - e^{\lambda(SD_{ti} - SL_{vmj})}, & SD_{ti} < SGL_{vmj} \end{cases} \quad (4)$$

The above equation (4) is used to calculate the possibility of process safety depending on the distribution of procedure t_i to VM is vm_j. If the virtual machine vm_j is designated to task, the constant t_i is set to 1, else it is set to 0.

$$P_{Security}(t_i) = \sum_{j=1}^{m} k_{ij}\, P_{security}(t_i, vm_j)/m \quad (5)$$

The proposed approaches, the possibility of privacy is determined by $P_{Security}(T)$ as described in equation (5) while evaluating the full operational task set T. $P_{Security}(T)$ determines the possibility of protection, ensuring that no tasks are vulnerable to assault throughout performance.

$$P_{Security}(T) = Ave\arg e\left(P_{Security}(t_i)\right) \quad (6)$$

6. Schedule primitives

The supplier in a cloud system has a huge amount of services, as well as the client needs to pay for them. Consider that the customer needs to purchase a quantity of virtual machines through the vendor. These virtual machines will represent various occurrence types. The amount of MIPs and cores determine the computing performance of virtual machines. We denote the computation capacity of each VM as $\omega(vm_j)$. The set of VMs is represented by $VM\ (vm_1, vm_2, ..., vm_m)$. Similarly each task has its own computation cost and it is designed as $\omega(t_i)$ for the task t_i. The execution time of the task t_i on the j^{th} VM of k^{th} type is termed as $ET(t_i, vm_j)$ and it is defined in equation (7).

$$ET(t_i, vm_j) = \frac{\omega(t_i)}{\omega(vm_j)} \quad (7)$$

The average processing time of the task t_i is given in equation (8)

$$\overline{PT(t_l)} = \frac{\sum_{j=1}^{m} PT(t_i, vm_j)}{m} \quad (8)$$

The processing cost of an edge $e(i, j)$ can be calculated using equation (9)

$$PC(t_i, t_j) = late(vm_j) + \frac{w(e(i, j))}{bw_k} \forall 1 \le j \le m,\ 1 \le i \le n \quad (9)$$

The average processing cost of edge $e(i, j)$ can be calculated as follows;

$$\overline{PC(t_i, t_j)} = \frac{\sum_{j=1}^{m} late(vm_j)}{m} + \frac{w(e(i, j))}{\sum_{k=1}^{0} bw_{k/k}} \forall 1 \le j \le m,\ 1 \le i \le n \quad (10)$$

The earliest start time (EST) of a process on a virtual machine is the latest feasible beginning period. It can be used to calculate in equation (11)

$$EST(t_i, vm_j) = \max\left\{\text{Re}\, ady\ time(t_i, vm_j)\right\}, \max_{t_m \in parent(t_i)}\left\{EET(t_m, vm_j) + PC(t_m, t_i)\right\} \quad (11)$$
$$\text{Re}\, dyTime(t_{entry}, vm_j = 0)$$

The earliest end time (EET) of a process in VM is the shortest duration it takes to complete it. This can be calculated by Eq. (12).

$$EET(t_i, vm_j) = EST(t_i, vm_j) + \frac{\omega(t_i)}{\omega(vm_j)} \forall 1 \le i \le n, 1 \le j \le m \quad (12)$$

TPT is the total table length obtained using the EET of the exit task and calculated using Eq. (13)

$$TPT = Makespan = EET(t_{exit}, vm_j) \quad (13)$$

TPC is the total cost of processing workflows using cloud resources. TPC is the quantity due to the cloud service by the client. Equation (14) is used to compute the TPC of Sequence,

$$TPC = \sum_{j=1}^{m} Cost(vm_j) * \frac{\left[\text{Re}\, leaseTime(vm_j) - \text{Re}\, adyTime(vm_j)\right]}{\tau} \quad (14)$$

7. Scheduling using enhanced flower pollination algorithm

The main objective of proposed methodology is to optimally schedule the task on resources by using EFP algorithm. For scheduling, we consider the three parameters such as execution time, cost and security. The overall structure of proposed methodology is given in figure 1.

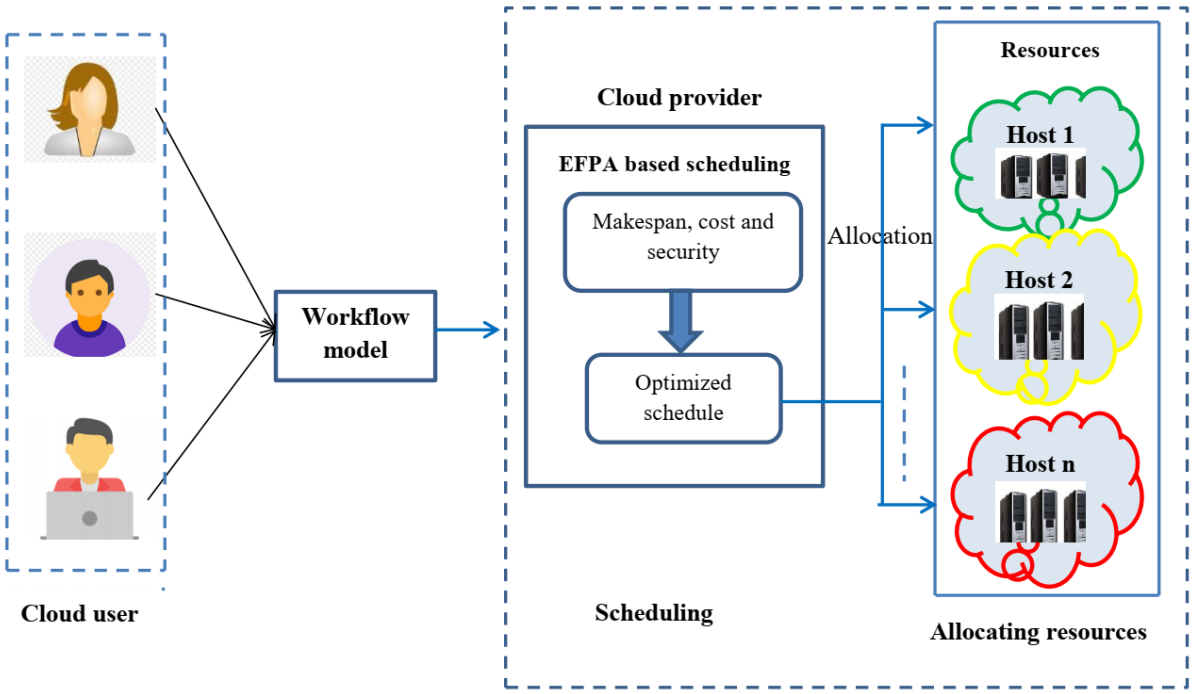

Figure 1: Task scheduling on cloud

Step 1: Solution encoding: Solution encoding is an important process for scheduling. In this paper, we consider four dimensions for encoding process. The particle consists of i^{th} task of work flow, VM id, virtual machine type and SGL of the VM. The SGL is calculated using equation (1).

Step 2: Fitness calculation: After the solution initialization, we calculate the fitness of each solution. The fitness is based on execution time, cost and security of the network.

$$F = Min\{\omega * OEC + (1-\omega) * OET\}$$
$$Subjected to\, SD_{t_i} \le SGL_{VM_j} \quad (15)$$

Step 3: Updation using EFPA

After the fitness calculation, each flower is updated with the help of EFPA. In FPA, two types of pollination are available namely, global pollination and local pollination. To enhance the performance of FPA, local pollination is replaced with the help o firefly algorithm. The global pollination can be represented mathematically as (16).

$$F_i^{t+1} = F_i^t + \gamma L(\lambda)\left(G_* - F_i^t \right) \quad (16)$$

Where, F_i^t is the pollen i or resolution vector F_i at emphasis t, and is the present best solution found among all solutions at the present age/cycle. The progression size is constrained by a scaling operator γ. Here, $L(\lambda)$ means the Levy flight-based advance size that relates to the capacity of pollination. The limited pollination and flower consistency can be demonstrated to as

$$F_i^{t+1} = F_i^t - \beta_0^{at}\, e^{-\gamma \ddot{D}_{ij}^2}\left(F_j^t - F_i^t\right) + \sigma_t \mu_i^k \quad (17)$$

The above equation is taken from the firefly algorithm [27]. In that overhead equation, F_i^{t+1} demonstrates the fresh updated solution, F_i^t illustrates the present i^{th} solution and F_j^t performs the j^{th} solution. Moreover, σ_t shows the arbitrary factor and μ_i^k is a random number and γ is the constant value.

Step 4: Termination criteria: The algorithm stops its performance when the best fitness value is selected. Once optimal fitness is achieved, the resource allocation is addressed.

8. Results and discussion

This section discusses the results and discussion of the proposed EFPA for task scheduling in cloud computing. The proposed task scheduling is done on an Intel Core i5 processor, and on a computer with 6GB of memory using the Windows 10 operating system. Simulation of the proposed method is implemented in JAVA. For experimentation analysis, three types of workflows are used such as Montage, CyberShake and LIGO. To prove the effectiveness of the proposed approch, we compare our algorithm with different algorithms namely, Heterogeneous Earliest Finish Time (HEFT) [27], Particle swarm optimization (PSO) [28] and Flower pollination algorithm [30].

Size	Workflow model	HEFT	PSO	FPA	EFPA
Small	Montage	230	210	150	75
	CyberShake	350	325	284	110
	LIGO	283	265	210	85
Medium	Montage	156	143	110	43
	CyberShake	444	420	360	210
	LIGO	399	374	310	190
Large	Montage	325	311	264	220
	CyberShake	650	625	575	320
	LIGO	555	541	495	310

Table 2: Performance analysis based on makespan

The performance of proposed approach is analysed based on makespan for different workflow such as Montage, Cybershake and LIGO. In this three types of VM instance small, medium and large are analysed. When analysing table 2, for montage workflow attained the minimum makespan compared to other two workflows. To prove the efficiency of the proposed EFPA based task scheduling, we compare our work with different task scheduling methods namely, HEFT [27], PSO based scheduling [28], FPA based scheduling [29] and EFPA based scheduling. The results show that proposed approach attained the minimum makespan for all the VM instance and workflows compared to other methods. This is due to EFPA based optimal scheduling. This method avoids the local optima by using levy flight strategy.

Number of tasks on workflow	HEFT	PSO	FPA	EFPA
500	700	630	570	310
1000	1800	1500	1200	950
1500	2400	2000	1700	1300
2000	3300	2700	2200	1800

Table 3: Performance analysis based on makespan by varying task size

In table 3, the performance of proposed approach is analysed based on makespan by varying task size. This is a synthetic dataset. When analysing table 2, proposed EFPA attained the minimum makespan of 1800sec for scheduling 2000 tasks, which is 2200s for FPA based scheduling, 2700s for PSO based scheduling and 3300s for HEFT based scheduling. It is clear from the table that makespan also increases as the task levels increases.

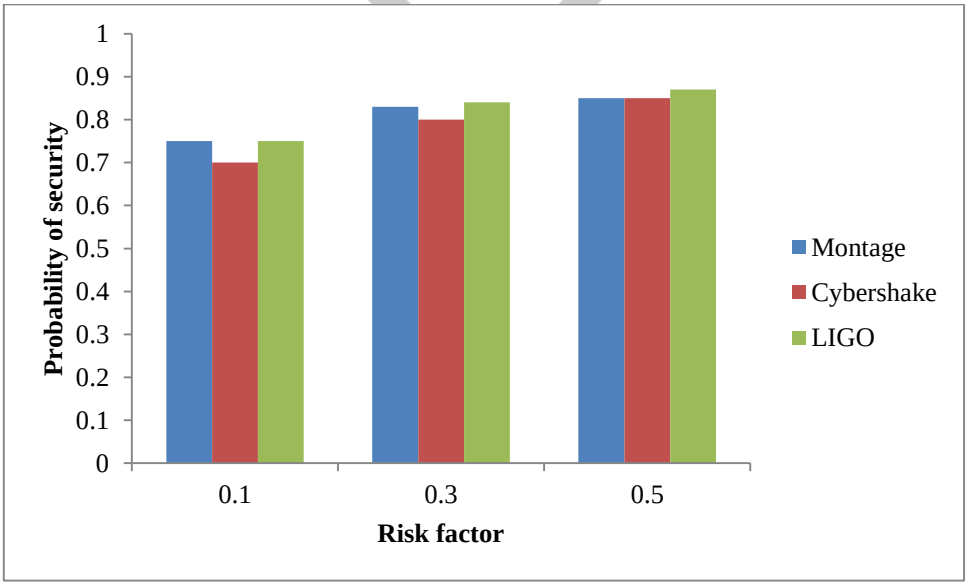

Figure 2: Probability of security by varying risk factor (λ) for small workflow application

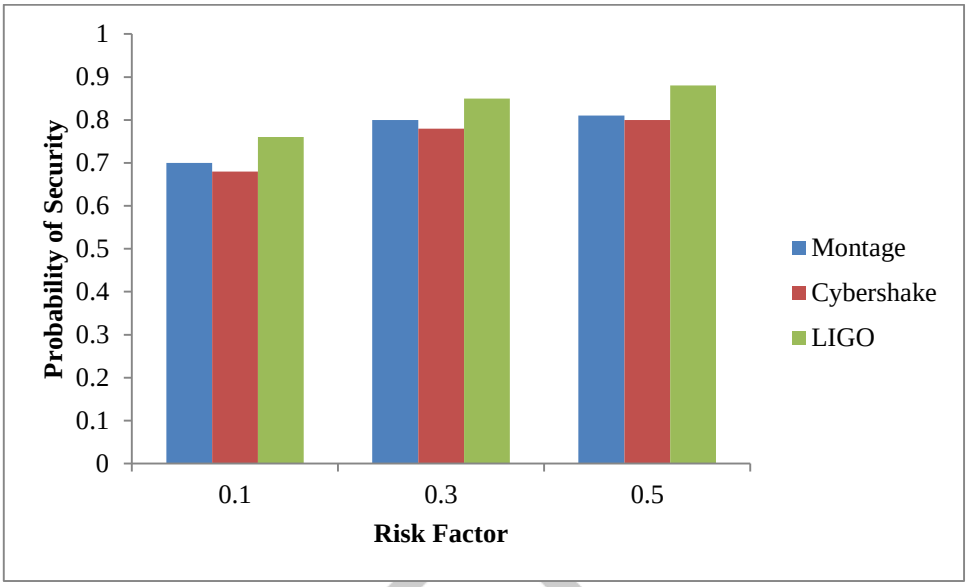

Figure 3: Probability of security by varying risk factor (λ) for medium workflow application

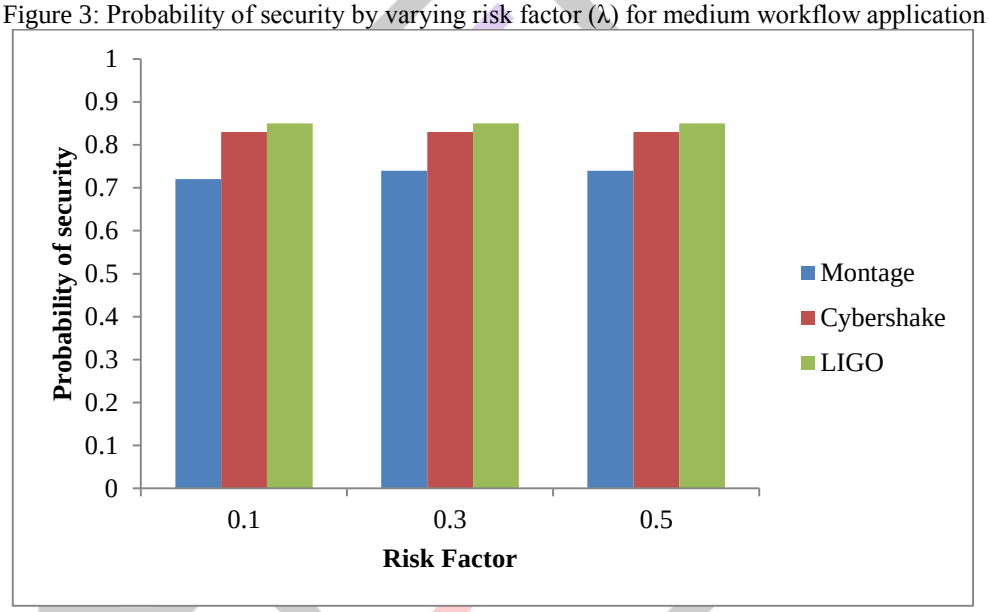

Figure 4: Probability of security by varying risk factor (λ) for large workflow application

In this paper, security is an important parameter for scheduling. The scheduling derivations are clearly explained in above sections. For security analysis of proposed approach, we vary the risk coefficient factor λ. The figure 2-4 shows the probability of security level using different VM instance namely, small, medium and large respectively. The proposed model ensures increased safety when the risk factor increases. Our system assigns VMs with the best security level. This is demonstrated with the probability of planned protection in Figure 2-4. The model thus proposed creates a good secure schedule for workflow applications in cloud computing.

9. Conclusion

An efficient workflow scheduling based on makespan, cost and security has been explained. To achieve this concept EFP algorithm has been explained. The flower pollination algorithm has been enhanced by levy flight strategy; this was a reason for achieving the maximum security level compared to other methods. To validate the proposed task scheduling technique, three workflow models has been used. Each workflow proposed method achieved maximum output compared to other techniques. In future, we will concentre fault tolerance strategy with task scheduling.

References

1. Amer, D.A., Attiya, G., Zeidan, I. and Nasr, A.A., 2022. Elite learning Harris hawks optimizer for multi-objective task scheduling in cloud computing. The Journal of Supercomputing, 78(2), pp.2793-2818.
2. Shanmugavadivel, G., Gomathy, B. and Ramesh, S.M., 2021. An Enhanced Data Security and Task Flow Scheduling in Cloud-enabled Wireless Body Area Network. Wireless Personal Communications, 120(1), pp.849-867.

3. Yang, J., Jiang, B., Lv, Z. and Choo, K.K.R., 2020. A task scheduling algorithm considering game theory designed for energy management in cloud computing. Future Generation computer systems, 105, pp.985-992.
4. Jena, R.K., 2017. Task scheduling in cloud environment: A multi-objective ABC framework. Journal of Information and Optimization Sciences, 38(1), pp.1-19.
5. Li, W., Fan, Q., Dang, F., Jiang, Y., Wang, H., Li, S. and Zhang, X., 2022. Multi-Objective Optimization of a Task-Scheduling Algorithm for a Secure Cloud. Information, 13(2), p.92.
6. Wei, X., 2020. Task scheduling optimization strategy using improved ant colony optimization algorithm in cloud computing. Journal of Ambient Intelligence and Humanized Computing, pp.1-12.
7. Srichandan, S., Kumar, T.A. and Bibhudatta, S., 2018. Task scheduling for cloud computing using multi-objective hybrid bacteria foraging algorithm. Future Computing and Informatics Journal, 3(2), pp.210-230.
8. Alworafi, M.A., Dhari, A., Al-Hashmi, A.A. and Darem, A.B., 2017. Cost-aware task scheduling in cloud computing environment. International Journal of Computer Network and Information Security, 9(5), p.52.
9. Krishnadoss, P. and Jacob, P., 2019. OLOA: based task scheduling in heterogeneous clouds. International Journal of Intelligent Engineering and Systems, 12(1), pp.114-122.
10. Li, W., Cao, S., Hu, K., Cao, J. and Buyya, R., 2021. Blockchain-Enhanced fair task scheduling for cloud-fog-edge coordination environments: Model and algorithm. Security and Communication Networks, 2021.
11. Arul Xavier, V.M. and Annadurai, S., 2019. Chaotic social spider algorithm for load balance aware task scheduling in cloud computing. Cluster Computing, 22(1), pp.287-297.
12. Gupta, A., Bhadauria, H.S. and Singh, A., 2021. Load balancing based hyper heuristic algorithm for cloud task scheduling. Journal of Ambient Intelligence and Humanized Computing, 12(6), pp.5845-5852.
13. Potluri, S. and Rao, K.S., 2020. Simulation of QoS-based task scheduling policy for dependent and independent tasks in a cloud environment. In Smart Intelligent Computing and Applications (pp. 515-525). Springer, Singapore.
14. Masadeh, R., Sharieh, A. and Mahafzah, B., 2019. Humpback whale optimization algorithm based on vocal behavior for task scheduling in cloud computing. International Journal of Advanced Science and Technology, 13(3), pp.121-140.
15. Basu, S. and Anand, A., 2019. Location based secured task scheduling in cloud. In Information and Communication Technology for Intelligent Systems (pp. 61-69). Springer, Singapore.
16. Ramegowda, A., Agarkhed, J. and Patil, S.R., 2020. Adaptive task scheduling method in multi-tenant cloud computing. International Journal of Information Technology, 12(4), pp.1093-1102.
17. Gabi, D., Ismail, A.S., Zainal, A., Zakaria, Z. and Al-Khasawneh, A., 2018. Hybrid cat swarm optimization and simulated annealing for dynamic task scheduling on cloud computing environment. Journal of Information and Communication Technology, 17(3), pp.435-467.
18. Prem Jacob, T. and Pradeep, K., 2019. A multi-objective optimal task scheduling in cloud environment using cuckoo particle swarm optimization. Wireless Personal Communications, 109(1), pp.315-331.
19. Zhu, Q.H., Tang, H., Huang, J.J. and Hou, Y., 2021. Task scheduling for multi-cloud computing subject to security and reliability constraints. IEEE/CAA Journal of Automatica Sinica, 8(4), pp.848-865.
20. Pradeep, K. and Jacob, T.P., 2018. CGSA scheduler: A multi-objective-based hybrid approach for task scheduling in cloud environment. Information Security Journal: A Global Perspective, 27(2), pp.77-91.
21. Kaur, P. and Mehta, S., 2017. Resource provisioning and work flow scheduling in clouds using augmented Shuffled Frog Leaping Algorithm. Journal of Parallel and Distributed Computing, 101, pp.41-50.
22. Fellir, F., El Attar, A., Nafil, K. and Chung, L., 2020, February. A multi-Agent based model for task scheduling in cloud-fog computing platform. In 2020 IEEE international conference on informatics, IoT, and enabling technologies (ICIoT) (pp. 377-382). IEEE.
23. Sharma, S. and Jain, R., 2019. EACO: an enhanced ant colony optimization algorithm for task scheduling in cloud computing. International Journal of Security and Its Applications, 13(4), pp.91-100.
24. Tawfeek, M.A. and Elhady, G.F., 2016. Hybrid algorithm based on swarm intelligence techniques for dynamic tasks scheduling in cloud computing. International Journal of Intelligent Systems and Applications, 8(11), p.61.
25. Abdullahi, M. and Ngadi, M.A., 2016. Symbiotic organism search optimization based task scheduling in cloud computing environment. Future Generation Computer Systems, 56, pp.640-650.
26. Xie T, Qin X (2006) Scheduling security-critical real-time applications on clusters. IEEE Trans Comput 55(7):864–879
27. Chopra, N., & Singh, S. (2013, July). HEFT based workflow scheduling algorithm for cost optimization within deadline in hybrid clouds. In 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT) (pp. 1-6). IEEE.
28. Wang, D., Tan, D., & Liu, L. (2018). Particle swarm optimization algorithm: an overview. Soft Computing, 22(2), 387-408.
29. Abdel-Basset, M., & Shawky, L. A. (2019). Flower pollination algorithm: a comprehensive review. Artificial Intelligence Review, 52(4), 2533-2557.